学校代码：10285

学　　号：20164227038

蘇州大學

SOOCHOW UNIVERSITY

硕士学位论文

（学术学位）

基于图像识别算法的物体认知系统关键技术研究及应用

Research and Application in Object Cognition System Based on Image Recognition Algorithm

研究生姓名	周靖越
指导教师姓名	姚望舒
专业名称	计算机科学与技术
研究方向	嵌入式与物联网
所在院部	计算机科学与技术学院
论文提交日期	2019 年 5 月

苏州大学学位论文独创性声明

本人郑重声明：所提交的学位论文是本人在导师的指导下，独立进行研究工作所取得的成果。除文中已经注明引用的内容外，本论文不含其他个人或集体已经发表或撰写过的研究成果，也不含为获得苏州大学或其它教育机构的学位证书而使用过的材料。对本文的研究作出重要贡献的个人和集体，均已在文中以明确方式标明。本人承担本声明的法律责任。

论文作者签名：周靖越　日　期：2019.6.23

苏州大学学位论文使用授权声明

本人完全了解苏州大学关于收集、保存和使用学位论文的规定，即：学位论文著作权归属苏州大学。本学位论文电子文档的内容和纸质论文的内容相一致。苏州大学有权向国家图书馆、中国社科院文献信息情报中心、中国科学技术信息研究所（含万方数据电子出版社）、中国学术期刊（光盘版）电子杂志社送交本学位论文的复印件和电子文档，允许论文被查阅和借阅，可以采用影印、缩印或其他复制手段保存和汇编学位论文，可以将学位论文的全部或部分内容编入有关数据库进行检索。

涉密论文□

本学位论文属　　在_____年___月解密后适用本规定。

非涉密论文□

论文作者签名：周靖越　日　期：2019.6.23

导 师 签 名：姚望舒　日　期：2019.6.23

基于图像识别算法的物体认知系统关键技术研究及应用

中文摘要

物体认知是指以嵌入式计算机为核心，利用传感器采集物体图像、形态、声音、气味等信息对物体进行识别的一种技术。传统的物体认知系统基于射频识别等传感技术，可快速进行物品追踪和数据交换，然而射频识别技术依赖于物品上的电子标签，有较高成本且有距离限制。随着硬件设备的升级以及网络技术的发展，图像逐渐成为了物联网大数据的组成部分。与此同时，针对图像数据的计算机视觉算法也得到了长足的发展。与纯粹的计算机视觉算法相比，物体认知系统更接近硬件设备与用户，与数据采集、传输以及交互过程结合的更加紧密。本文基于现有图像识别技术，从运算分布、数据传输、可增量学习等角度入手，研究了利用物联网设备采集到的图像进行物体认知所需的关键技术。本文的主要工作如下：

（1）针对物体认知系统的图像处理能力，提出了适用于物联网环境的增量图像识别框架。本框架将特征提取、训练、推理的过程切分开来，使系统具有快速持续学习的能力，并使得设备的计算能力可以得到更好的使用。

（2）针对物体认知系统的运算分配问题，提出了多层神经网络推理过程在多层物联网架构下的切分算法。与传统物体认知技术不同，在数据采集端完成图像识别的所有操作代价过于高昂。而物联网设备相互独立，设备间的数据传输会消耗资源。本文针对这一问题，以最快响应为目标，提出了运算分配算法，加速了推理过程。

（3）基于上述技术，设计了基于图像识别技术的物体认知系统，并给出了它在移动端与嵌入式设备上的实现。这是一套高自由度，广适应性的系统，与应用需求结合紧密，能以极低的代价为用户提供个性化的解决方案。

关键词：图像识别算法，深度学习，物体认知系统，物联网，嵌入式应用处理器

作　　者：周靖越

指导教师：姚望舒

Research and Application in Object Cognition System Based on Image Recognition Algorithm

Abstract

Object cognition is a technology that uses the image, shape, sound, smell and other data collected by the sensor to identify objects. Object cognition system usually bases on embedded computers. Traditional object cognition systems base on sensing technology, such as RFID, can quickly track object and exchange data. However, RFID technology relies on electronic tags on objects, which has high cost and distance limitation. Images have gradually become a part of the Big Data of the IoT with the development of hardware equipment and network technology. At the same time, the image recognition technology based on deep learning has made progress and has high recognition accuracy. Compared with the computer vision algorithm, the object recognition system is closer to hardware and users. Based on the existing image recognition technology, this paper focus on the techniques needed for object recognition using the images collected by IoT devices. This paper focuses on computing distribution, data transmission and incremental learning. The main work of this paper is as follows:

(1) This paper proposes an incremental image recognition framework suitable for the IoT environment. The framework separates the processes into feature extraction, training and inference. These three parts can be allocated to different IoT devices as independent modules. Reasonable allocation of these three modules allow the system to incremental learning and improve system performance.

(2) This paper proposes an algorithm to allocate the operation of neural network inferencing process into multi-layer IoT devices to accelerate object cognition. Different from traditional object cognition technology, image recognition algorithm needs more resources, which makes it too expensive to complete all the processing in the data acquisition device. However, IoT devices are independent, and data transmission between devices will

consume resources. This algorithm can obtain the optimal allocation strategy, which can speed up the object cognitive inference process deployed in the IoT environment.

(3) This paper design and implement an object cognition system on an embedded system based on the technology above. The object cognition system proposed is an open system with wide adaptability. Users can design the specific functions of the application. This highly interactive pattern can provide users with personalised solutions at a low cost.

Keywords: Image Recognition, Deep Learning, Object Cognition System, Internet of Things, Embedded Application Processor

Written by Jingyue Zhou

Supervised by Wangshu Yao

目 录

第一章 绪 论

物体认知是指以嵌入式计算机为核心，利用传感器采集到的信息对物体进行识别的技术。本文的物体认知系统基于图像识别算法，而图像数据量大，处理复杂，难以使用嵌入式计算机独立完成所有操作，因此需要借助物联网技术与服务器协同运算。本文围绕物体认知系统与图像识别算法结合的过程中存在的一系列问题展开研究。

1.1 研究背景

本文研究基于图像识别算法的物体认知系统。本节给出了物体认知系统的定义，阐述了本文研究的背景，以及本文的研究目的与意义。

1.1.1 物体认知系统概述

物体认知，即说出物体的名字，是幼儿学习启蒙的开端，蕴含了人类智能中的“示教、学习、识别”的基本过程。嵌入式物体认知系统是指以嵌入式计算机为核心，利用图像、声音、气味、形态等传感器采集物体信息，进行标记、训练与推理的系统。基于图像识别的嵌入式物体认知系统利用嵌入式计算机通过摄像头采集物体图像，利用图像识别算法进行训练，训练完成后，可进行推理完成对图像的识别。体现了人类智能中的“示教、学习、识别”基本过程，展示了人工智能中“标记、训练、推理”的基本要素。

图像数据的普及为物体认知的发展带来了新的可能，同时也带来了新的挑战。随着硬件设备的升级以及网络技术的发展，图像逐渐成为了物联网大数据的组成部分。与此同时，针对图像数据的计算机视觉算法也得到了长足的发展。图像数据具有适用性强，应用范围广，展示更为直观等优势。在工业，生活，教育等领域都有广阔的发展前景。但同时，在嵌入式设备上部署基于图像识别的物体认知系统也面临很多挑战。与传统数据不同，图像数据信息密度低，数据量大，处理过程繁琐。而嵌入式设备资源受限，通常无法独立完成所有操作，因此需要借助物联网技术与服务器协同运算。而在这样的一个包括了嵌入式终端，物联网传输，服务器在内的系统中，各个设备应

当完成什么工作，互相之间该如何高效配合等问题都需要探讨。这是一个杂糅了图像识别算法，物体认知，物联网传输技术，嵌入式应用处理器在内的交叉问题，需要结合算法的实现细节以及实际设备的可用资源进行考虑。而本文抽取了其中不变的，共性的部分，进行了物体认知系统的框架设计。

本文所提出的物体认知系统基于物联网通信技术。物联网是基于传统互联网基础上功能更加强大，应用更广泛的网络技术[1]。人们对于物联网的理解，已经从刚开始的传感器网络扩展到了万物互联。国际数据公司的分析表明，到了 2020 年，全球将有 300 亿智能终端接入互联网，这些设备将产生海量数据[2]。物联网吸引了越来越多的企业，投身于物联网基础设施建设、物联网设备通信、物联网数据处理以及基于物联网的系统应用开发等各个产业中。但是对于物联网未来发展所需要解决的关键技术以及物联网的演进路线人们还没有统一的认识。

本文所提出的物体认知系统是物联网的一种具体应用模式。对物联网应用的研究成为引导物联网发展方向的指路标。目前，对于物联网技术的研究着重于通信方面，研究物联网设备之间的通信协议与标准，例如窄带物联网[3]。此类研究为未来各场景下各设备间的低成本可靠连接打下了基础。然而，至今为止仍未出现物联网领域的标志性应用。虽然有智能抄表[4]，自动驾驶[5]等应用案例出现，但从其经济效益，产业影响力来说，还没有达到它应有的高度。通信技术的升级仅仅是搭建了物联网实现的平台，却不能为物联网的繁荣发展提供内在驱动力。而这驱动力的来源在于物联网应用的普及，所需的核心技术是物联网数据的理解与应用技术。

本文所提出的物体认知系统是人工智能技术与物联网技术的一次融合，它基于人工智能技术中的图像识别算法。人工智能的兴起为物联网提供了更多的应用可能。人工智能概念诞生以来，受智能算法、计算速度、存储水平等多方面因素的影响，人工智能技术和应用发展经历了多次高潮和低谷。近年来，以深度学习[6]为代表的机器学习算法在机器视觉和语音识别等领域取得了极大的成功，识别准确性大幅提升，使人工智能再次受到学术界和产业界的广泛关注。物联网技术作为数据采集的技术，为人工智能提供了大量的数据。而围绕云计算[7]，大数据[8]等技术进行的研究，保障了人工智能所必须的运算资源。人工智能的发展模式也从过去追求“用计算机模拟人工智

能”，逐步转向以机器与人结合而成的增强型混合智能系统，用机器、人、网络结合成新的群智系统，以及用机器、人、网络和物结合成的更加复杂的智能系统。

本文所提出的物体认知系统是物联网加人工智能方向上的一次具体尝试。目前物联网技术发展趋于成熟，研究的热点逐渐转向对于物联网技术的应用。人工智能技术具备大量的应用场景，例如图像处理，自然语言处理等。本文的物体认知系统研究了将可增量图像识别算法运用于物联网系统时的一系列关键技术，为智能物联网的发展做出了新的贡献。

1.1.2 研究目的和意义

本课题旨在融合物联网领域及深度学习领域的一些最新技术，对智能物联网领域持续学习问题，运算分配问题进行探讨，并提供一套有实际使用价值的智能物联网系统，争取为智能物联网体系的发展贡献新的思路。

本文所提出的物体认知系统针对智能物联网环境下的问题提出了解决方案。如前文所述，受物联网应用场景、系统架构、硬件设备、网络情况等条件的限制，智能物联网有诸如持续学习能力的缺乏以及系统响应不够及时等问题亟需解决。与增量学习技术的融合可以使智能物联网应用更加灵活，相较于传统的代码更新方案，通过自动进行的数据分析、参数更替等方法更新软件消耗的代价更小，反应速度也更快。流数据分析处理技术的应用可以大大降低系统的存储要求。特征提取技术在终端设备的部署则可以减少网络传输的数据量，同时降低数据传输所导致的安全隐患。合理的运算分配可以更好的调度系统拥有的资源，提升系统表现。诸如此类的技术研究与融合可以提升智能物联网的实际表现，扩展其所适用的范围，开拓新的应用领域。

本文所提出的物体认知系统是物联网加人工智能方向上的一次具体尝试。对智能物联网的研究出现的相对较迟，有较多的技术空白待填补。物联网本身是比较成熟的技术，经过多年的发展，在硬件设施、网络传输、大数据技术支持等方面已较为完善。人工智能技术历经了多次起伏，如今在图像处理、自然语言处理、语音处理等领域取得了长足的发展。深度学习作为热门的人工智能技术之一，近些年受到广泛关注，对于它的优势、劣势的研究已经相对透彻。但是对于两者之间的融合，也就是智能物联

网技术的研究相对较少。物联网技术的发展是个循序渐进的过程，硬件与软件的升级都不是一蹴而就的。早期的物联网设备的运算能力以及网络的传输能力不能满足复杂应用的需求，因此物联网系统也遵照终端收集数据、云端存储处理数据的模式。此时物联网、云计算、大数据、数据挖掘等技术被作为相对独立的分支学科进行研究。各自解决各自的技术难点，而跨领域的研究较少。随着技术的发展，物联网设备拥有的运算能力增强，在应用中合理使用这些设备的运算能力能带来不可忽视的好处。这才导致了边缘计算，雾计算等技术的兴起。直到这时，物联网技术和人工智能技术才被视为一个整体进行考虑。在设计物联网应用的架构时，需要考虑人工智能算法的需求、各部分的运算量、存储需求、以及设备需求。在应用人工智能算法时，也需要考虑到物联网系统本身的限制。可以说，智能物联网是一个相对新兴的、跨学科的技术。

1.2 国内外研究现状

本文所提出的物体认知系统结合了国内外对物联网架构、图像处理等方向的一系列研究成果。人工智能技术与物联网的结合一直是国内外研究的热点。本文的物体认知系统主要针对智能物联网应用的持续学习问题以及运算分配问题进行研究。本节围绕这两个问题，从智能物联网系统架构以及图像识别算法自身的发展入手，阐述了如今的国内外研究的现状。

1.2.1 智能物联网系统架构有关研究

传统的物联网架构以云为中心。从终端采集得到的原始数据被直接传送到云端，存储以及数据分析都在云端完成，处理后的结果则被传送回终端，如此完成一次数据循环。然而，这种以云为中心的物联网架构存在若干弊端。例如，大量的数据需要被传输到云端，从而导致了时间、电量等一系列开销。此外，这种方式需要在终端与云端之间长时间维持一条快速且稳定的通信线路，这在很多情况下都是不现实的。而且这样的原始数据传输也会带来安全隐患。

为了克服以云为中心的弊端，出现了在云外处理数据的物联网架构。随着技术的发展，终端设备具有了越来越强的处理能力。同时用户的需求也越来越多样化，在很多应用场景下，以云为中心的架构的弊端变得明显。此时雾计算[10]，边缘计算[11]等旨

在利用云外设备处理能力的技术应运而生。

有许多研究致力于探索如何将机器学习算法运算分配到物联网系统的设备中。近年来，深度学习作为机器学习算法的代表，受到了广泛关注。深度学习算法包括训练与推理两个组成部分。训练为从已有数据中提取有效信息的过程，而推理为利用提取出的有效信息处理待处理数据的过程。通常训练过程所需的运算量远大于推理过程，而对于一个系统而言，调用推理过程的次数远大于调用训练过程的次数。

其中，一部分研究侧重于对推理过程的分配。普遍来讲，深度学习训练过程消耗巨大，远超终端设备的负载能力，即便对于云端来说，训练依然是一个耗时耗力的工作。此外，由于物联网数据持续生成的特性，物联网系统很难获得全局视野。云端都无法获得一个完整的，稳定的训练数据集，对于终端来说，视野就更为狭窄。出于上述原因，这部分研究设定模型在系统部署前就已经训练完成，且在系统运行的过程中不会再进行训练操作。在此条件下，他们研究如何将推理过程部署到物联网系统中，以达到低延时、低功耗、高可靠性、高安全性等目的。这些研究主要针对物联网终端运算能力弱，无法脱离云端独立完成推理任务与传输数据到云端代价高这一矛盾展开。有的研究对模型进行了一系列压缩，简化工作，并成功将它部署在了以嵌入式设备为载体的终端设备中[12][13]，这种形式移除了云端的负载，推理过程不再依赖网络，使得应用的表现更加稳定。然而对于另一些应用来说，综合利用整个系统的运算能力可以取得更好的效果，一些研究专注于此，以神经网络推理过程易于分割为切入点，将推理运算分配到整个系统中[14][15]。Teerapittayanon 等人[16]则进行了调整网络结构以适应物联网多层架构的研究，将多出口网络结构应用在了物联网环境下。

为使系统能持续运行，对部署后更改模型能力的研究也受到了重视。系统需要跟上外部环境以及用户需求的变化，需要在部署之后依然拥有更新自己的能力。顾会光等人[17]针对传统无线远程代码更新时数据大量丢失的问题，提出了一种分段式接收远程代码并集中更新的设计方案。除了对远程更新技术本身的研究以外，也有研究者关注于深度学习的特性。Song 等人[18]提出了在终端筛选数据，只将有价值的数据传输给云端进行训练的方法。Shokri 等人[19]则侧重于联合学习所导致的安全隐患，他们提出了一种随机传送梯度与参数更新的方法以保护隐私。还有很多对于深度学习本身

的研究，可以解决物联网环境带来的挑战，本文将在下一节阐述。

1.2.2 图像识别算法有关研究

对于数据挖掘算法的改良可以加速智能物联网的发展。深度学习技术是热门的图像识别方法。近年来，以深度学习为代表的各类人工智能技术发展迅速。作为从数据中挖掘，运用有效信息的手段，这些技术也被智能物联网应用广泛使用。作为一种机器学习方法，深度学习有很多优势，也有很多的局限。在深度学习与物联网的融合过程中，这些优势与局限也展露了出来，很多针对深度学习技术自身的研究对于智能物联网的发展也大有帮助。

对深度学习模型的精简可以改善物联网设备计算能力弱的困境。受制于成本及功耗的限制，需要精简在物联网终端设备上进行的运算。对于深度学习模型而言，有向更大更精确方向的研究，也有向着小而精简方向的研究。近年来有很多面向移动设备的深度学习部署运算优化技术出现[20]。Han 等人[21]对训练好的深度学习模型中权重较小的连接进行了剪枝，该方法大大减少了模型参数数量。Gupta 等人[22]采用了降低权值存储精度的方法，实验表明，模型精度损失较小，而运算速度和功耗表现则大幅提升。此外，在一些深度学习应用，特别是图像类应用中，卷积过程占推理过程运算量的比重很大，因此也存在针对卷积层的优化方案[23][24]。

对增量学习的研究可以使智能物联网应用获得跟上环境变化的能力。对于传统的深度学习算法而言，它在一个数据集上进行训练之后便会记得这个数据集中包含的知识。之后如果在另一个数据集上再次训练这个已训练过的模型，它会学习到新数据集中的知识，同时忘记从旧数据集学习到的知识。这个现象被称为灾难性遗忘[25]，它是深度学习技术的固有特性，无法被完全克服，但可以通过一系列技术手段[26][27][28]削弱它带来的负面影响，这些技术手段被统称为增量学习。对于智能物联网应用而言，灾难性遗忘的特性会带来很大麻烦。物联网数据是持续产生且一直变化的，需要应用不断从新产生的数据中获得知识。同时物联网产生的数据又很庞大，应用不可能永远保留全部的历史数据作为训练数据集。这就使智能物联网应用很难在不忘记历史数据中知识的前提下学习新数据中的知识，而融入增量学习相关技术可以使智能物联网在这个问题上有更好的表现。

无监督学习技术可以帮助处理物联网产生的大数据。大数据具有数据规模大，数据类型繁多，数据流转速度快，数据密度低的特点[29]。对于大数据的存储及使用一直是热门的研究方向，涉及到了文件系统，数据库系统，索引与查询，数据分析等方方面面。对于深度学习技术而言，传统的深度学习是有监督的学习方法，需要在已标注的数据集上进行训练。但是物联网产生的数据大多没有被标注过，这使得人们难以利用深度学习直接从物联网数据中提取有效信息。然而，利用无标注的数据训练深度学习模型也一直是研究者关注的方向[30][31]，运用这些技术可以更方便的从物联网数据中提取有效信息。

迁移学习技术则可以有效减少物联网系统的运算量，降低应用的开发难度。对于一些深度学习应用，例如图像类应用而言，从头开始训练模型需要耗费很大的代价。但其实模型内的很多的信息是通用的，即使数据集或者用户的需求发生改变，这些信息仍然有重复利用的价值。以一个已经训练好的模型为基础构建新的应用能节省很多精力，也能更容易获得好的应用表现。这一类技术被称为迁移学习[32]，能使训练获取的信息得到跨应用的使用[33]。物联网与现实密切相关，其环境复杂，用户需求多样。迁移学习技术可以帮助打破不同环境、不同应用之间的开发壁垒，加速智能物联网的发展与落地。

1.3 物体认知系统针对的问题

本文所提出的物体认知系统针对了智能物联网的一系列问题。由于物联网本身具有的一些特性，在智能物联网发展的过程中，有很多的问题需要解决[9]。例如：

对持续学习的能力的需求。物联网系统会持续的运行，数据会持续生成，这就要求对应的数据处理技术拥有处理连续数据流的能力。持续生成的数据意味着没有边界的存储需求，在很多情况下，系统无法保留所有的历史数据。此外，在物联网系统运行的过程中，外界环境与用户需求都会发生改变，系统需要具备自主适应这些变化的能力。由于数据是持续生成的，没有尽头，系统难以获得全局的视野。而且物联网数据大而驳杂，有效数据密度较低。传统数据挖掘技术运行在一个精细而完整的数据集上，这与物联网持续生成数据的环境有很大不同，将这些技术融入物联网的环境需要

克服很多难题。

终端设备的固有局限。终端设备有电池电量的限制，运行在终端设备上的程序需要最小化能量消耗。此外，物联网终端通常数量巨大，受成本等因素的限制，终端设备的运算能力也较弱。另外，与用户的交互也是一个问题。在物联网环境下，用户一般通过终端设备与系统进行交互。然而相较于个人电脑等设备，终端设备可以进行交互的手段更为匮乏。

设备间数据传输的限制。物联网终端是数据的生成方，需要通过无线网络进行数据传输，而无线网络相对而言不够稳定，而且带宽也会受到限制。在一些具有高实时性需求的应用中，设计者需要考虑到传输的延时。此外，终端的数量巨大，终端与服务器通信时需要考虑到并发性等技术难点。而在数据传输的过程中，安全与隐私也是需要考虑的问题。

1.4 研究内容

智能物联网的研究中诸如持续学习能力的缺乏，以及系统响应较慢等问题值得进一步研究。本文针对这两个问题，结合了国内外对物联网架构、图像处理等方向的一系列研究成果，设计了一套可增量图像识别框架，提出了深度学习推理过程在多层物联网架构下的切分算法，并实现了一款基于图像识别技术的物体认知系统。

针对物体认知系统的图像处理能力，提出了适用于物联网环境的可增量图像识别算法框架。不同于传统智能物联网框架，这是首次将深度学习中的增量学习算法与物联网环境进行的融合。由于深度学习训练过程运算量巨大等原因，当前试图在智能物联网环境下进行深度学习训练的研究较少。然而流数据是物联网的一大特性，持续学习能力也是智能物联网一大应用的需求。本文融合了增量学习，迁移学习等深度学习技术，针对物联网环境设计了一套可增量图像识别框架。该框架分为特征提取、增量训练、实时推理三个模块，这三个模块可以根据实际系统情况以及应用需求分别被部署在终端设备、服务器、或者其他物联网设备中。为了减少运算量需求，适应物联网环境，本框架将整个图像识别模型一分为二。前半部分是特征提取部分，该部分输入的是传感器采集到的原始图像，输出的是提取出的特征。特征提取部分模型参数通用

性较强，可以从其他通用任务中迁移而来，这一操作可以降低应用的开发难度与成本。此外，特征提取部分在部署之后不再改变，不需要对这部分的参数进行再训练，这一特性可以降低系统设计难度以及后期维护成本，尤其是当特征提取部分部署在资源受限的终端设备上时。固定特征提取部分参数的设计还减轻了后期再次训练时的运算负担，使实时的学习以及结果反馈成为可能。后半部分是增量学习部分，该部分的输入是提取出的特征，输出是应用相关的分类结果，该部分是应用持续学习能力的保障。对该框架的实验表明，这样设计的增量学习算法仍具有相当高的分类准确度。

针对物体认知系统的运算分配问题，提出了多层神经网络推理过程在多层物联网架构下的切分算法，加快了物体认知过程。深度学习技术基于多层神经网络，而多层神经网络具有层间运算相互独立的特性。基于这个特性深度学习比较容易被拆解成多份，分配到不同的设备中。本文根据物联网运算能力分布、网络传输特性等情况，以最短响应时间为目标在多层物联网架构下设计了深度学习推理过程切分算法。传统物联网技术遵照终端与服务器端的两层架构，此前对于深度学习推理过程运算分配的研究也局限于这样的两层架构。直到边缘计算，雾计算等技术的兴起，才有了对多层物联网架构的研究。本文首次将这一技术扩展到了多层物联网架构中，根据运算量，设备运算能力、数据传输量、网络传输速度等关键信息，建立了模型，并提出了最小化响应时间的切分算法。可以从理论上证明，本文提出的切分算法可以取得本文建立的多层智能物联网模型下的最优解，实验也证明了这一结论。

基于上述技术，依照可封装、易移植的模块化思想，设计并实现了一款基于图像识别技术的物体认知系统，验证并展示了本文工作的可行性与有效性。该系统是本文提出的可增量图像识别框架的具体应用。本系统可以被部署在由移动端与服务器端组成的典型移动应用环境中，也可以被简化并部署在单一的嵌入式设备中。本系统允许用户建立各自的模型并通过自己上传的已标注图片以增量的方式训练模型，然后使用训练出的模型对自己上传的图片进行分类。传统的智能物联网应用往往针对于某一个特定领域，例如人脸识别，工业故障检测等，这样的方法能针对这个领域达到较高的准确率，但是缺乏广泛的适用性。本文设计的系统针对娱乐、生活等准确度要求相对较低的应用领域，以适当降低准确度为代价获得了更强的应用适用性。

1.5 论文结构

本文对基于图像识别算法的物体认知系统关键技术进行了研究与应用。本文针对智能物联网持续学习能力缺失，运算分配不合理等一系列问题进行了研究。论文主要内容如下：

第一章：绪论。本章阐述了本文的研究背景以及本文所针对的问题，罗列了如今针对这些问题的研究现状以及这些研究的优势，然后阐述了本文的研究目的与研究意义，最后给出了本文的结构。

第二章：物体认知系统相关技术。本章阐述了与本文所研究的物体认知系统相关的一系列技术，分析了这些研究各自的优势，陈述了这些研究与本文提出的物体认知系统之间的关系。

第三章：可增量图像识别框架。本章阐述了本文所设计的可增量图像识别框架。该框架可以被分为三个相对独立的模块。本章先阐述了框架的整体情况，各模块之间的关系，然后给出了各模块的实现细节，最后针对该框架的各模块进行了实验。

第四章：多层神经网络推理过程切分算法。本章建立了深度学习推理过程在多层物联网架构下的运算分配模型，然后在该模型下以最小化响应时间为目标提出了一种多层神经网络推理过程切分算法，最后进行了实验以验证该方法的实用性与有效性。

第五章：物体认知系统设计与实现。本章描述了本文所实现的物体认知系统，先阐述了整体框架，再描述了该系统各部分的细节，然后通过若干实验直观的展示了系统运行效果，并罗列了若干可以应用该系统的场景，最后讨论了系统的资源占用情况。

第六章：低资源嵌入式设备上的实现。本章依照本文提出的框架，对各部分做了大幅简化，提出了一种极低资源消耗的图像识别算法，对该算法进行了测试，并给出了它在Cortex-M系列嵌入式设备上的实现。

第七章：总结与展望。本章总结了本文的工作，并提出了今后可能的发展方向。

第二章 物体认知系统相关技术

本文设计的物体认知系统使用了基于深度学习的图像识别算法，是深度学习与物联网环境的融合。本章将对物体认知系统所涉及到的深度学习算法作简要概述，并阐述如今的各种物联网架构，以及一些融合深度学习与物联网应用的研究，以此奠定本文所做工作的技术背景。

2.1 相关深度学习技术

本文所提出的物体认知系统所使用的图像识别算法主要基于深度学习技术[6]。深度学习架构，如深度神经网络、深度置信网络、以及递归神经网络，已应用于计算机视觉、语音识别、自然语言处理等领域，它们产生的结果可与人类专家相媲美，在某些情况下能优于人类专家。本节阐述了深度学习技术中，与物体认知系统相关的迁移学习技术与增量学习技术，以及这些技术在物体认知系统中的应用。

2.1.1 迁移学习

在深度学习中，每层神经网络都能将输入的数据转换为更抽象与更复合的表示[34]。大多数现代深度学习模型都基于人工神经网络，深度学习中的深度，指的是数据进行转换的层数，在不存在递归神经元等结构的情况下可以被理解为神经网络的层数。虽然理论上层数大于 2 的神经网络就已被证明是一种通用逼近器，可以用来模拟任何函数。但是深度模型与深度较少的模型相比通常具有更好的表现，这是因为额外的神经网络层有助于学习特征。例如，在图像识别应用中，原始的输入可能是像素矩阵，第一层神经网络可能被用来抽取出图像中各种边缘的信息，第二层可能被用来记录各种边缘的排列，识别出各种形状，第三层则可能被用来识别出更具体的形状，例如鼻子和眼睛，第四层则可能被用来识别出面部。深度学习技术可以在训练过程中自动的学习到特征应当分布在哪一层。通常通用的特征会位于低层，而任务相关的特征会位于高层。在使用深度学习进行有监督图像分类任务时，研究者发现，无论使用什么图像数据集，模型的第一层都像是一个通用的滤波器[35]。也就是说，无论面对什么数据，

模型的第一层都会执行一些通用的计算过程，这些过程并不仅针对某个特定的数据集或者特定的任务，而是通用而普遍的。模型的后几层则慢慢的趋于任务相关，会因为数据集以及任务的变化有较大的改变，即失去了这种通用性。直到模型的最后一层，输出的分类结果会与任务完全相关。

低层的神经网络具有良好的任务无关性，把从一个任务中训练好的模型的前几层网络迁移到另一个任务中可以取得良好的效果，这样的技术被称为是迁移学习[32]。当目标数据集的规模显著小于基础数据集时，迁移学习是一个可以用来训练大型网络达到较好的分类效果却又不会导致过拟合的工具。在 Yosinski 等人[36]的研究中有着对迁移学习过程的描述。通常进行迁移学习的方法是将已经训练好的基础网络的前几层复制到目标网络的前几层中，对于目标网络较高层的参数则进行随机初始化。然后使用目标数据集训练目标网络。在训练目标网络的过程中，对于迁移过来的低层网络有两种不同的处理方式。第一种方式是仍用反向传播算法传递误差来训练这些低层网络中的参数，这种方式被称为是微调。第二种方式是固定迁移过来的低层网络参数，在接下来的训练过程中不再对它们做出改变，这种方式被称为是冻结。对于这两种处理方式的选择往往视目标数据集的大小以及迁移来的参数数量而定。如果目标数据集很小而迁移来的参数很多，进行微调往往会导致参数过拟合，此时应当使用冻结的方法。反过来，如果目标数据集较大而迁移过来的参数较少，那么过拟合的情况就不会很严重，此时使用微调的方法可以提高模型的表现。当然，如果目标数据集非常大，那么进行迁移学习的意义就不大了，因为使用新数据集就足以训练低层网络的参数。

物体认知系统中使用了迁移学习的技术，模型的低层网络是从通用图像分类任务的网络中迁移过来的。在物体认知系统框架中，这部分迁移过来的网络被称为是特征提取部分。在再训练的过程中，对于这部分迁移过来的参数使用了冻结的操作。物体认知系统具备可持续学习能力，它需要不断从用户提供的新数据中学习新知识。而每次用户新提供的数据的数量都不会很大，不足以支持对整个网络进行再训练。此外，对迁移过来的部分进行再训练就意味着将这部分网络向着特定的任务上拟合，这也意味着这部分网络会丧失通用性，转而只针对于这一个特定的任务。物体认知系统面临的是多变的物联网环境，而不是某一个具体的应用。在物体认知系统中，特征提取部

分作为一个通用的数据预处理模块而存在，在部署后的训练过程中它始终保持着冻结，它不会为某一类用户指定的应用而设计或进行改动，这样的设计思路也使其拥有了更良好的通用性以及可持续学习能力。不再对特征提取部分进行微调还降低了再次训练所需的资源，加快了再训练的速度，为用户提供了更好的使用体验。在物体认知系统中，特征提取部分被部署在数量最多，分布最广泛的终端设备上。在部署后的的运行过程中不再对特征提取部分进行更改这一特性可以降低系统的复杂度以及后期维护成本。

2.1.2 增量学习

持续学习能力的缺失一直是深度学习技术所面临的难题之一。训练神经网络的过程，是使用方向传播算法不断修正网络参数，直到网络参数收敛到固定的值的过程。当网络训练完成后，网络中的参数便代表了它从输入数据中抽取出的信息，以及从输入数据中找到的一种匹配模式。此时输入给网络与训练数据相似的数据，网络便可以使用它学习到的匹配模式输出相应的结果。然而，当这个已训练完毕的网络在一个新的数据集上进行训练后，它便会忘记曾在旧数据集上学习到的知识，这个问题被称为是灾难性遗忘[25]。当新数据集与旧数据集的差距较大，乃至于新数据集中完全不包含旧数据集中的类别的数据时，灾难性遗忘的问题便会更加明显。这个问题是由神经网络技术，或者说是反向传播算法本身所导致的。反向传播算法将对模型中所有的参数向能降低损失的方向进行调节。而一般来说，损失就是网络输出结果与训练数据的标注之间的差异。也就是说，使用反向传播算法进行训练的目的仅仅是拟合训练数据。在这个过程中并不存在任何能使网络记住已习得知识的约束。而反向传播算法执行的又是如此的彻底，为了更好的拟合训练数据，改动了所有需要改动的参数直到结果收敛。因此一轮训练过程结束后，剩下的只是拟合了新数据集的网络，与旧数据集基本没什么关系了。

持续学习能力很重要。从应用的角度来讲，真实世界的环境是动态的，会随着时间不断改变。在一些应用场景下，当系统被训练完毕，部署下去后，外部环境仍会发生改变。若要使系统跟上这种变化，同时具备原有的能力，那么传统的方法只能将原有的数据集与新采集到的数据混合，然后在这个更大的新数据集上训练模型。这样的

方法存在弊病，随着系统的运行，为了保留所有信息，数据集势必越来越大，这会给系统造成越来越大的负担。

虽然在深度学习系统中，持续学习能力缺失的问题不能被完全解决，但是仍存在一系列被称为增量学习的技术手段可以削弱这个问题带来的影响。之前提到过，灾难性遗忘出现的原因之一是在使用反向传播算法训练模型的过程中不存在能使网络记住已习得知识的约束。反向传播的目标是使损失降低，那么只要更改损失函数，使损失函数中包含记住已习得知识的倾向，便可以在一定程度上保留已习得的知识[27]。反向传播的结果是拟合训练数据集，如果训练数据集中包含了所有类别的数据，那么在训练完成后，模型仍会记住所有类别的信息。保留所有的数据会快速的增加系统的负担，但若通过一些技术手段，从原始数据中抽取出一部分重要的数据作为样例保留，在每次训练时将样例与新数据混合作为训练数据集，就能在保证系统负担不快速增加的同时记住所有类别的信息。Rebuffi 等人[28]提出的增量学习方法主要就是基于这种保留样例的方法。反向传播算法会改动所有的参数来拟合训练数据，导致这些参数中蕴含的旧有知识无法得到保存。若通过一些技术手段，限制对记住旧知识较为重要的参数在新训练过程中的改动，就能在新训练过程结束后，让模型仍然保留一部分旧的知识[26]。

物体认知系统中运用了以保留样例为主的增量学习技术。物体认知系统的设计宗旨是以低代价为用户提供自定义的图像识别解决方案。在部署完成之前，物体认知系统无法获知用户的具体任务需求。具体的任务需求是在系统的运行过程中由用户逐步教给系统的。也就是说，物体认知系统遵循着先部署，后针对用户自定义的任务进行训练的流程。具体到模型结构上，物体认知系统的高层网络中运用了增量学习技术。物体认知系统会通过选取样例特征的方式，为所有已习得的类别保留一定数量的样例数据，每次系统收到新的训练数据时，会将新数据与样例数据混合作为训练集。具体的实现方法将在 3.2 节阐述。与上述针对增量学习本身的研究不同，物体认知系统着眼于增量学习技术在物联网环境下的部署与使用。物体认知系统对增量学习模型进行了切分与更改，使模型中包含的运算能更好的被分配到物联网硬件设备中，并达到更好的系统性能。

2.2 物联网相关技术

本文提出的物体认知系统是针对物联网环境而设计的。本节阐述了物联网相关的各种概念，物联网发展的历程，以及如今物联网的特性。

2.2.1 物联网概述

物联网指的是将各种信息传感设备，如射频识别装置、红外感应器、全球定位系统、激光扫描器等种种装置与互联网结合起来而形成的一个巨大网络。其目的是让所有的物品都与网络连接在一起，系统可以自动的、实时的对物体进行识别、定位、追踪、监控并触发相应事件[37]。传统物联网的关键技术包括标签事物、识别事物的射频识别技术[38]，集分布式信息采集、信息传输和信息处理技术于一体的无线传感网络技术[39]，包括人机交互、智能控制、智能信号处理在内的智能技术，以及与微缩元器件有关的纳米技术。经过多年的发展，物联网在智能电网、智能交通、智能物流、智能家居等领域获得了广泛应用[40]。

2.2.2 物联网与云计算

物联网在被提出之后，逐渐与云计算技术融合起来。物联网技术包括射频识别技术、无线传感网络技术等一系列涉及到数据采集与传输的技术，却并未涉及到处理与使用这些数据的技术。由于物联网应用对可靠性、高性能、高效率和可扩展性的需求，在一些场景下，以服务器为中心的物联网架构不再合适。因此出现了结合物联网与云计算的研究[41]。云计算的概念意味着可以用在因特网上托管着的资源、服务、以及技术在需要的时候进行组合以提供更复杂的服务。云计算有着按需提供服务、联网即可访问、池化的资源、极佳的弹性与可扩展性等特点[42]。云计算的这些特性对于一些物联网应用来说至关重要。例如，池化的资源提高了服务的可靠性与效率，按需提供服务以及弹性是高效率与可扩展性的基础。物联网负责采集数据，云计算则帮助其克服处理和存储限制，这种高效的配合使这两者同时得到快速发展。

然而，以云为中心的物联网架构仍然存在一些缺点[11]。第一，是网络传输能力的限制。由于云端的计算能力超越了其余物联网设备的计算能力，将所有的计算任务放在云端已被证明是一种有效的数据处理方式。然而，与快速发展的数据处理速度相比，

网络的带宽已经停滞不前。随着物联网终端生成的数据量不断增加，数据传输速度成为了以云为中心的物联网应用的瓶颈。例如，在自动驾驶的应用中，车辆每秒能生成千兆字节量级的数据，并且需要实时的处理这些数据才能做出正确的决策。如果将所有车辆产生的数据发送到云端进行处理，则响应时间会太长。而且若一个区域内存在大量进行着数据传输的车辆，网络的带宽与可靠性都会受到影响。在这种情况下，就需要在终端设备上处理数据以缩短响应时间，提高处理效率并降低网络压力。第二，是物联网数据的增长超过了云计算的能力[43]。物联网终端的数量增长迅速，将在几年内增长到数十亿以上，这些终端设备所产生的原始数据将是巨大的。现有的云计算能力不足以处理所有的这些数据，这意味着需要在终端设备上处理部分数据。第三，是云为中心的物联网架构不能充分利用逐渐发展的终端设备的能力。传统的物联网数据产生者是一系列搭载着各式传感器的嵌入式设备。这些设备运算能力有限，只具备基础的数据采集与网络传输功能。如今，随着基础设施以及应用需求的发展，如智能手机等设备也逐渐担任起物联网数据生产者的角色。这些设备会采集大量图片、视频等信息密度较低的数据，而这些设备本身则具备一定的数据处理能力，在这些设备上先对数据进行一些预处理操作可以有效降低云端的负担。此外，如智能手机、可穿戴设备等终端采集到的数据通常是私有的，在终端设备上处理这些数据能够更好的保护用户的隐私。

2.2.3 物联网与边缘计算

以云为中心的物联网架构的缺陷导致了边缘计算的兴起[11]。边缘计算指允许将计算部署在网络边缘的技术。这里的边缘被定义为在数据源与云服务器之间的任何计算和网络资源。边缘计算的基本原则是计算应该在数据源附近进行。边缘计算与雾计算[10]是类似的概念，但边缘计算更侧重于利用终端设备的运算能力，而雾计算更侧重于利用基础网络设施的计算能力。

在物体认知系统的设计过程中借鉴了边缘计算的思想。物体认知系统在终端设备上部署了特征提取模块，当终端设备接收到图像后，会先对图像进行特征提取操作，然后将特征上传以进行后续操作。这个特征提取操作可以减轻网络传输的负担，并分担服务器端的计算任务。此外，多层神经网络推理过程切分算法针对的也是包含了终

端设备、网络设备在内的多层物联网环境。

2.3 深度学习与物联网的融合

本文所提出的物体认知系统是结合了基于深度学习的图像识别算法与物联网环境的研究。物联网是采集数据的技术，深度学习是使用数据的技术。这两种技术本相互独立，却又逐渐相互融合形成了若干交叉学科。针对深度学习与物联网的融合，本节阐述了融合深度学习与物联网的意义、在融合过程中浮现出的问题、国内外针对这些问题的研究、以及在这些研究的基础上本文所做的工作。

2.3.1 融合深度学习与物联网的意义

将深度学习技术引入物联网应用可以更好地利用物联网传感器采集到的声音，图像等数据。移动互联网设备以及各种嵌入式设备的激增产生了一个充斥着丰富信息的世界。近年来，市场上出现了一系列智能物联网产品，例如各类的智能家具[44]。这些产品能捕获环境数据，并使用结构化的方式将这些数据记录下来，然后使用各类传统的数据分析技术来分析这些数据。此外，还有例如语音识别设备，人脸识别设备等一些物联网设备需要采集非结构化的多媒体数据，例如音频信号和图像数据。大部分的传统机器学习技术不具备处理这些数据的能力。此时便需要引入深度学习这种擅长处理这些非结构化的多媒体数据的技术。

在物联网应用中部署深度学习解决方案通常遵循着先训练，后部署的流程。深度学习算法包括训练与推理两个组成部分。训练是从已有数据中提取有效信息的过程，而推理是利用提取出的有效信息处理待处理数据的过程。使用深度学习技术的模式是，先在一个足够充足的数据集上对模型进行训练，然后部署使用训练得到的模型。训练一个大型深度学习模型需要耗费大量的资源。当将深度学习技术应用到物联网环境中时，通常先在拥有庞大运算能力的服务器中完成训练过程，再将训练过的模型部署在物联网设备中。

2.3.2 在融合过程中需要解决的问题

在将训练得到的模型部署到物联网设备中的过程中会遇到一些问题[45]。一些物联

网设备上运行的深度学习应用程序，例如自动驾驶、智能监控，具有严格的实时要求。如果将这些推理运算分配在云端，一来势必要增加由于传输数据所带来的延时，二来需要始终如一的优质网络连接，这限制了应用所能部署在的地理位置，并导致了更高的成本。若将推理运算分配在移动设备端，那么应用的表现就不会受网络连接的影响。然而，在移动设备上实现深度学习很困难。在硬件方面，移动物联网设备的一大设计需求是低功耗[46]，这通常意味着有限的计算能力以及存储空间。而深度学习模型通常包含了较多的参数，需要消耗一定的存储资源。除了训练过程，深度学习的推理过程也需要大量的运算并且具有高功耗。在软件方面，为了减少内存占用，移动物联网设备上的应用程序通常不使用操作系统，或者使用非常轻量级的操作系统[47]和最少的第三方库。然而现有的深度学习框架通常会调用到很多的第三方库，而这些库很难移植到物联网设备中。

2.3.3 融合深度学习与物联网技术的相关研究

为了在物联网环境中使用深度学习模型进行推理运算。针对于上述问题，研究者们展开了一系列工作，主要朝着推理运算在物联网环境中的分配方案，以及深度学习模型的精简两个方向展开。

一些研究针对推理运算在物联网环境中的分配方案展开。这些研究中包括了一些将推理运算全部部署在终端设备上的研究。例如，Cheng 等人[48]在配备了集成 GPU 的手机上进行了人脸识别运算。在他们的实验中，单幅图像的识别时间为秒级。Soyata 等人[49]提出了多层移动云混合架构，该架构使用具有 GPU 加速的中间设备将面部图像转换成特征图，然后将特征图传输到云端进行人脸识别的最终处理。借助这种中间层辅助架构，他们成功地减少了整体处理延迟，满足了实时性的要求。Teerapittayanon 等人[16]进行了调整网络结构以适应物联网多层架构的研究，将多出口网络结构部署在了物联网设备中。这一研究可以根据当前结果的置信度，在合适的时候结束推理运算，从而降低资源消耗。深度学习模型具有分层的特点，其推理运算能够被切分开来。针对切分深度学习推理运算以部署在物联网设备上的问题。Li 等人[15]建立了深度学习推理运算在终端-云两层物联网环境下的运算分配模型，并以加速响应为目标提出了切分算法。

另一方面，也存在针对深度学习模型本身以及其实现方式进行优化的研究。在软件方面，如果可以充分的利用现有的计算组件，则程序可以获得更好的性能。Sun 等人[50]使用 ARM Compute Library 对深度学习模型的推理运算进行了测试，这是一个针对 ARM 平台进行了高度优化的运行库，能够获得更快的执行速度。TensorFlow 框架[51]中也增加了 TensorFlow Lite 模块，这是一个针对移动端推理运算的开源框架，它能为物联网移动设备上的深度学习推理运算的部署提供方便。与硬件架构有关的另一种技术是模型压缩，这里的压缩主要指降低模型参数的存储精度。对于深度学习模型本身，研究人员已经发现，如果使用较低精度的模型参数，仍然可以达到几乎相同精度的结果[52][22]。降低参数精度最直接的好处是可以降低存储开销，如果使用 16 位或 8 位参数而不是 32 位，则内存开销可以减少 50%或 75%。此外，通过硬件的配合，降低参数精度还可以带来更快的执行速度。对于深度模型本身，也有着努力使其变得更加轻巧的研究。Han 等人[21]依照仅学习重要的连接的思路提出了一种模型修剪方法。通过进一步优化和权重共享，他们对模型进行了高于 10 倍的压缩，并提高了 3 倍以上的执行速度。此外，在一些深度学习应用，特别是图像类应用中，卷积过程在推理过程中运算量的占比很大，因此也存在针对卷积层的优化方案。Howard 等人[24]基于流线型架构，提出了深度可分离卷积以构建轻量级深度神经网络。这一设计大大减少了模型的尺寸以及推理过程所需的资源。

物联网环境下应用的持续学习能力，即随环境变化而更新的能力也受到了研究者的重视。物联网系统运行在不断变化的外部环境中。通常一个深度学习应用在部署前已经在足够大型的数据集上进行了训练。然而在一些应用场景下，设计者仍然无法保证应用所运行的环境一直与数据集所来源的环境保持一致。这时候就需要重新训练模型，并且对已部署的设备进行更新。针对无线远程代码更新问题，顾会光等人[17]提出了一种分段式接收远程代码并集中更新的设计方案，该方案可以减少更新时的数据丢失。也有研究者着眼于物联网应用部署后，获得可用于更新模型的新训练数据的方法。Song 等人[18]提出了在终端筛选数据，只将有价值的数据传输给云端进行训练的方法。该方案针对了物联网数据价值密度低的特点，能有效的降低系统负担。网络传输导致的安全问题也一直受到研究者的重视。Shokri 等人[19]侧重于联合学习所导致的安全隐

患，提出了一种随机传送梯度与参数更新的方法来保护隐私。然而，在物联网持续学习能力方面，还存在着数据集持续增长，训练资源消耗大，参数更新机制不完善等问题。

2.3.4 物体认知系统所做工作

本文所提出的物体认知系统中应用了上述对运算分配方案以及深度学习模型的研究。受 Soyata 等人[49]的启发，物体认知系统借鉴了多层物联网架构的思想，将运算过程分成了三个相对独立的模块，并能根据物联网终端设备、网络设备、云服务器等各自的特性进行部署。在 Li 等人[15]的基础上，本文的第四章也针对深度学习推理过程在多层物联网框架下的部署问题进行了研究。此外，在物体认知系统实现的过程中，使用了 Han 等人[21]提出的轻量级网络结构以及 TensorFlow Lite 框架。

本文所提出的物体认知系统是针对于物联网应用的持续学习能力而设计的。物体认知系统中应用了基于样例筛选与保留的增量学习技术[28]。这一技术能够有效抑制训练数据集的增长，同时保留训练数据集中的重要信息。物体认知系统中应用了冻结特征提取网络的操作，这一操作减少了再次训练时需要更新的参数数量，从而减少了再次训练时消耗的资源，加快了再次训练的速度。此外，在物体认知系统所使用的可增量图像识别框架中，部署在物联网终端设备上的特征提取模块是固定不变的，在系统的运行过程中不需要对其进行更改。这一设计免去了对物联网终端设备的更新，降低了系统设计与实现的复杂度，同时节省了运行过程中更新终端设备所需的资源消耗。

2.4 本章小结

本文所提出的物体认知系统是部署在物联网环境下的，基于深度学习技术的可持续学习图像识别系统。本章阐述了基于深度学习的图像识别技术基础，与物体认知系统相关的物联网，云计算，边缘计算等概念，这些概念之间的渊源，以及其发展的来龙去脉。最后阐述了融合深度学习与物联网的意义，在融合过程中浮现出的问题，国内外针对这些问题的研究，以及在这些研究的基础上本文所做的工作。

第三章 可增量图像识别框架

随着物联网硬件设备与传输能力的发展，图像识别技术被广泛应用到了各类物联网系统中。然而如深度学习一类的计算机视觉算法缺乏持续学习的能力，不能很好地适应动态变化的物联网环境。针对这一问题，本章提出了一种可增量图像识别框架，它基于物联网框架，并且融合了一系列增量学习最新技术，使基于图像识别的物体认知系统具备了持续学习的能力。

3.1 系统概览

本节提供了本章提出的可增量图像识别框架的整体描述。本节分为三个小节。第一小节描述了为实现增量图像识别而提出的模型整体结构。第二小节描述本框架对于这个模型的分割，以及如何将被分割开的模型塞进物联网架构中。第三小节提供了本章要处理的问题的形式化描述。

3.1.1 适用于智能物联网的模型结构

图 3-1 展示了分类网络的整体结构，这一小节将简略描述这个结构。整个模型包括了两个部分：特征提取部分与增量学习部分。特征提取部分的作用包括压缩数据，去除数据中的敏感信息，分担运算等。特征提取模块可以经由以下步骤建立。第一步是在一个大型图像分类数据集上对一个卷积神经网络进行充分的训练。第二步是移除网络末端与分类任务相关的全连接层。第三步是冻结剩余网络的参数，将它作为特征提取网络。第 3.2 节将对这一过程做详细的阐述。用于图像分类的卷积神经网络的前几层类似于边缘，纹理，形状的筛选器[36]，这些内容相对任务独立，因此具有很强的通用性。冻结的部分在部署之前就已经训练完毕。这种训练是单次的，在系统开始运行之后不必再花时间来训练特征提取网络。这意味着在特征提取网络的训练上可以投入充足的精力，可以花费大量时间在一个大型数据集上训练特征提取网络。因此特征提取网络会经受充足的训练，可以处理各种不同种类的图像。然而，仅仅只有这一部分模型不具备持续学习的能力。增量学习部分能够带来这种能力。增量学习部分是两层结合了样例选取，蒸馏损失，以及最近邻分类等增量学习最新技术的全连接层。第 3.2 节将对增量学习部分做详细描述。

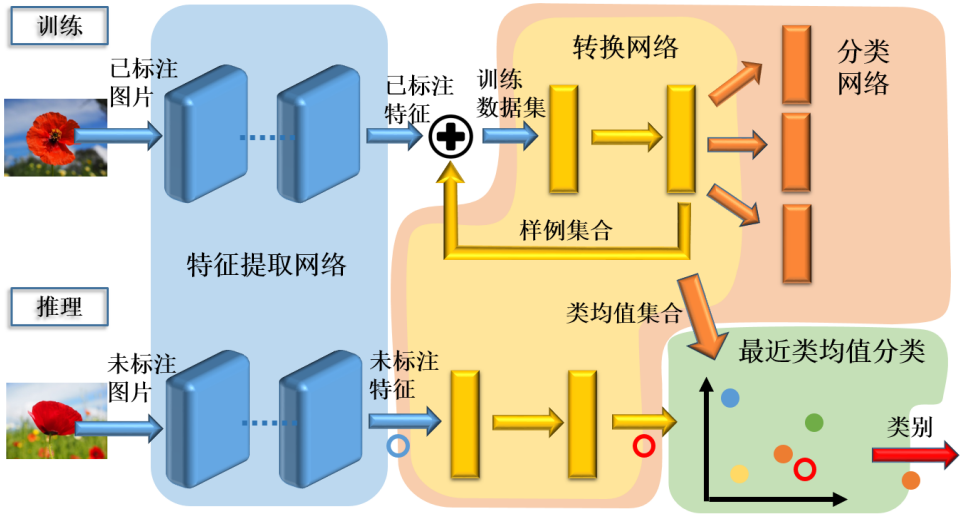


图3-1 推理过程运算分配模型

3.1.2 分布式部署

这一小节将介绍整个网络在物联网设备中的部署，图 3-2 展示了这种部署。整个模型被切分成三块，分别是特征提取部分，增量训练部分，以及实时推理部分。在本章的架构中，这三个部分分别被分配在终端设备，服务器，以及网络设备中。

特征提取部分被部署在终端设备中。在本章的架构中，终端设备是整个系统与外界的交流渠道。每当终端设备从外部环境中采集到图片时，终端设备将从这图片中提取出特征，并将提取出的特征传输到网络设备中。由于特征的大小通常小于图像本身的大小，这一步骤能够有效地压缩数据，进而减少数据传输量。在终端设备中进行特征提取操作有助于增强传输过程的安全性，有助于用户数据的隐私保护。在终端进行特征提取的操作保证了网络中没有原始图像的传输，只有提取出的特征。特征提取过程丢失了大量与应用无关的信息，此外，从特征中提取出敏感信息的难度也大大增加。另外这个特征提取过程分担了一定的运算量，在特征的基础上进行操作能降低后续过程的运算量。这个特征提取部分在部署后不会再改变，可以将它封装成输入为图像，输出为特征的模块写死在终端设备中。这一特性能降低整个智能物联网应用的后期维护负担，尤其是当终端设备数量巨大，分布广泛而分散时。

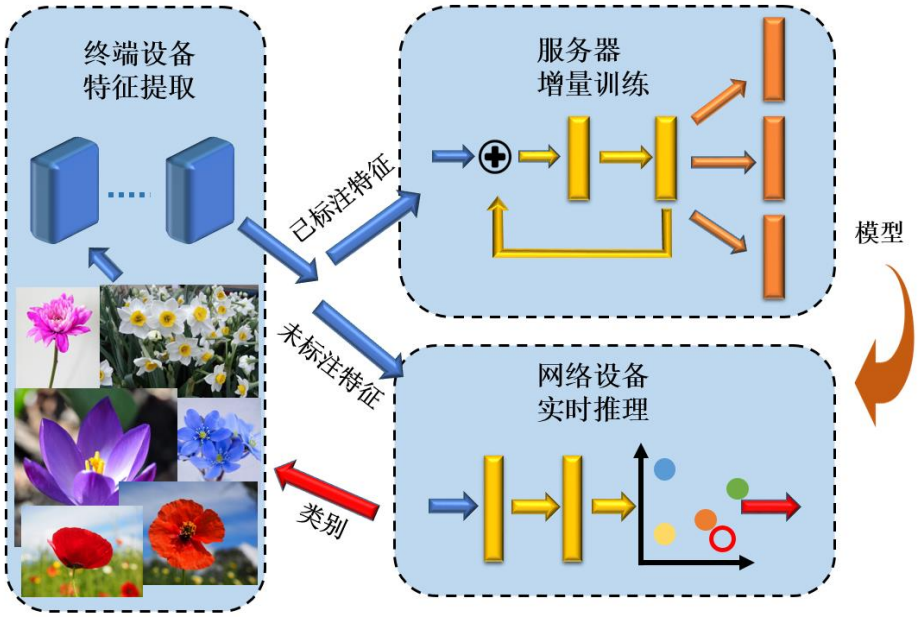


图3-2 推理过程运算分配模型在物联网环境下的部署

增量训练部分被部署在服务器上。每当系统接收到有标注的图片时，先在终端提取图片的特征，再将带标注的特征传输到服务器端进行增量训练。每当需要更新分类模型的时候，服务器端会将模型传输到网络设备中。更新的时机视应用而定，可以由用户发起，也可以设置为定时更新。服务器端被视为与终端传输耗时相对较高，运算能力相对较强的设备。训练过程消耗的资源相对较多，而对实时性的要求则相对较低。因此将训练部分部署在服务器上。

实时推理过程被部署在网络设备上。在本章的框架下，网络设备指的是如物联网转发节点等具有一定计算能力，能耗要求相对较低，位于终端节点与服务器之间的设备。推理过程所使用的模型由服务器训练而来，会随着系统的运行而发生改变。此外推理过程的实时性要求相对较高，因此被部署在离终端节点相对较近的网络设备中。

当然，对模型的部署并不是一成不变的，模型的这三个部分可以被灵活的部署。在很多现实情况下，物联网应用仍然遵循着终端到服务器的两层架构。此时可以根据应用的需求灵活部署。如果应用中的终端是运算能力较强的设备，而且系统需要具备离线工作的能力，例如终端是一个车载应用，需要在没有网络的情况下仍有一定工作能力。便可以将特征提取，推理过程部署在终端，将训练过程部署在服务器上，此时的终端设备将具有离线工作的能力。如果应用中的终端运算能力较弱，而且软件更新较为麻烦，例如终端是一款搭载在低功耗嵌入式芯片上的为某个应用单独设计的摄像

头。则可以将特征提取部分部署在终端，将推理，训练过程部署在服务器端，此时的终端设备将省去参数更新的麻烦。

3.1.3 框架描述

就特征提取过程而言，需要从原始图像 x 中提取出特征 f ，即

$$f = \text{Extraction}(x) \tag{3.1}$$

就增量训练过程而言，系统需要不断从新的已标注数据中学习知识。也就是说模型 M 可以不断从已标注数据中习得知识，即

$$\text{M} = \text{Train}(\text{M}, f^{y}) \tag{3.2}$$

其中 f^{y} 代表了被标注为类别 y 的特征。就实时推理过程而言，系统需要不断使用已习得的知识对无标注特征进行分类，即

$$y = \text{Inference}(\text{M}, f) \tag{3.3}$$

其中的 y 即为模型将特征 f 分到的类别。在 3.2 节中，将对这三个过程，以及模型 M 的具体内容进行详细解释。

3.2 具体实现方法

本节提供了本文提出的可增量图像识别框架的详细实现。本节分为三个小节。第一小节描述了部署在终端的特征提取方法。第二小节描述了部署在服务器端的增量训练模块。第三小节描述了部署在网络设备上的实时推理方法。

3.2.1 特征提取

特征提取过程的作用是压缩数据，减少数据传输时的安全隐患，以及分担运算。特征提取的公式可以表示为

$$f = \sigma(x) \qquad (f \in \square^{p}) \tag{3.4}$$

σ 是特征提取网络。特征提取网络的作用是将原始图像 x 转化为特征 f 。特征 f 是一个 p 维实数空间中的点。p 是由系统设计者设定的参数，p 同时也是特征提取网络 σ

的最后一层神经元个数。p 越大则单个特征的数据量越大，这导致需要从终端设备上传的数据量变大，同时单个特征可能包含的信息量也变大。对 p 的取值可以根据具体应用需求决定。特征提取网络σ是在系统部署前就确定了的，在部署后不会对其进行任何更改。而σ的效果是抽取出原始图像中的有效信息，σ的能力直接影响了后续的训练，推理等工作。因此σ需要具有较强的通用性，需要在大型的图像数据集上进行长时间的训练。而正是由于σ的通用性，它可以从其他通用图像识别应用中迁移而来。此外，σ被部署在终端设备上，对其运算量需做一定控制，可以考虑应用各种削减运算量的方案。

3.2.2 增量训练

这一小节将介绍增量训练方法。增量训练即从不断接收到的新数据集D中学习新知识刷新模型M 的过程。受[27][28]启发，增量训练方法结合了样例集合选取，蒸馏损失，以及最近类均值分类等技术。样例集合选取以及蒸馏损失是在训练过程中使用的，在这一小节中将进行详细介绍。最近类均值分类在推理过程中使用，这一部分内容将在 3.2.3 节详细介绍。

这一段中将阐述本章使用的记号。模型M 中包含了特征提取网络σ，分类网络ϕ，转换网络φ，样例集合E，以及类均值集合R。也就是说，$\mathrm{M}=\{\sigma,\phi,\varphi,E,R\}$。其中特征提取网络$\sigma$被部署在终端设备，在系统部署完成后不会再发生改变。分类网络ϕ，转换网络φ，样例集合E，以及类均值集合R会随着训练的过程而不断更新。推理过程只需要用到转换网络φ以及类均值集合R。事实上，转换网络φ就是分类网络ϕ的一部分，而类均值集合R可以由样例集合E计算得到。因此，其实只需要特征提取网络σ，分类网络ϕ，以及样例集合E就能包含模型M 中的所有信息。但为了简化叙述，以及准确的对运算分配，数据传输等过程的描述，模型引入了转换网络φ和类均值集合R的概念。在 3.2.1 节中已对特征提取网络σ进行了介绍。分类网络ϕ由若干层全连接层组成，以特征f为输入，以标注y为输出。分类网络ϕ是从系统不

断接收到的新数据中训练得到的。转换网络φ是将分类网络ϕ最末端用于输出分类结果的全连接层去除后得到的。转换网络φ以特征f为输入，以转换后的特征$\hat{f}$为输出，即

$$\hat{f}=\varphi(f) \qquad (\hat{f}\in\square^{q}) \tag{3.5}$$

$\hat{f}$是一个q维实数空间中的点。与p一样，q也是一个自定义的参数。q的值变大则会导致单个样例的数据量变大，这会加大增量训练模块的存储以及运算开销，但是也能保留更多的信息，一定程度上能提高分类准确率。f是通用的图像特征，它与用户定义的任务关系不紧密，也不会随着增量训练的过程而改变。而$\hat{f}$则不同，$\hat{f}$是应用相关的。这是因为转换网络φ是通过用户标注的数据训练出来的，是应用相关的。而转换网络φ的作用就是将应用无关的f转化为应用相关的$\hat{f}$。$\hat{f}$之间的距离对于应用有着实际的意义，即，应用中相同类别的图像生成的$\hat{f}$之间的距离较近，而应用中不同类别的图像生成的$\hat{f}$之间的距离较远，这里的距离指欧氏距离。这个特性是后续特征选取以及最近类均值分类的基础。样例集合$E=\{e_1^1,...,e_j^i,...,e_m^n\}$，其中$e_j^i$表示被标注为类别$i$的第$j$个特征。$e_j^i\in\square^{p}$，即每个样例都是一个带标注的转换前的特征。$e_m^n$表示总共有$n$个类别的图像，且第$n$个类别包含$m$个特征。值得注意的是，每个类别包含的特征数可以不同。每当增量学习模块收到已标注的特征集时，增量学习模块会将这个已标注的特征集合与样例集合混合，再使用这个混合数据集训练分类网络ϕ。样例集合包含了所有类别的数据，因此这个混合数据集中也包含了所有类别的数据，在这个混合数据集上训练得到的分类网络也就会记得所有类别的信息。可以说，样例集合E是所有历史数据的浓缩。由于神经网络存在灾难性遗忘的问题，分类网络ϕ本身不具备长效记忆的能力，这才需要有样例集合E来留存历史信息。类均值集合$R=\{r^1,...,r^i,...,r^n\}$，其中$r^i\in\square^{q}$。用$e^i$来表示所有属于类别$i$的样例集合，则$r^i$为$e^i$中所有样例经过转换网络$\varphi$后的均值，即$\overline{\hat{e}^i}$。类均值集合$R$在推理过程时使用。

这一段中将阐述样例集合选取过程。样例集合 E 需要能代表所有的历史数据。如果用 $\hat{F}$ 代表所有数据在 $\mathbb{R}^q$ 中的映射，则选取 E 时的评价标准为

$$\min \sum_{i=1}^{n} \left\| \overline{\hat{e}^i} - \overline{\hat{f}^i} \right\| \tag{3.6}$$

上式的意思是使得 E 经过转换网络后得到 $\hat{E}$ 中的属于各类别的特征的均值与 $\hat{F}$ 中分属于这些类别的特征的均值的距离的和最短。值得注意的是，样例集合 E 中的数据是没有经过转换网络 φ 的，这样 E 才能被用来训练分类网络 ϕ。然而，对于 E 的选取标准则经过了转换网络 φ，在转换过后的空间 $\mathbb{R}^q$ 中。这是因为对于 E 的选取是一个应用相关的过程，只有在经过了转换网络 φ 之后，特征之间的距离才有了应用相关的含义。样例集合 E 中的样例的数量受到限制，不得超过一个自定义的容量 v。容量 v 将会被均匀的分配给每一个已知类别。v 的值设置的大，则保留的样本数量多，模型所能容纳的类别数量就大，且能一定程度的提高准确率，但是会增大增量训练模块的存储消耗。样例集合的选取过程如算法 3-1 所示。对于每一个类别来说，如果该类别数

算法 3-1：样例集合选取方法

Input : 数据集合 $F=\{f^1,...,f^n\}$，转换网络 φ，样例集合容量 v

Output: 样例集合 $E=\{e^1,...,e^n\}$

$\hat{F}=\varphi(F)$，$ve=v/n$，$E=\{\}$

for $i \in 1,...,n$ **do**

  if $ve >= |f^i|$ **then**

    Append f^i to E

  else

    $e^i=\{\}$，$m=\overline{\hat{f}^i}$，$w=m$

    for $i \in 1,...,ve$ **do**

      $k=\arg\min_{\hat{f}_k^i \in \hat{f}^i \cap \hat{f}_k^i \notin e^i} \left\| \hat{f}_k^i - w \right\|$ //找到与所需特征最相似的特征

      Append f_k^i to e^i

      $w=w-\hat{f}_k^i+m$

    Append e^i to E

return E

据量小于等于各类别的样例容量ve，则该类别的所有数据都可以作为样例保留。若超过了样例容量，则每次选取加上后能使该类已选取的样例转换后的均值最接近于该类所有数据转换后的均值的数据加入该类别的样例，直到选满ve个。

特征选取的步骤可以在一定程度上减弱不平衡数据集所带来的影响。这一步骤为每个类别保留了相同数量的样例，不论在原始数据中这些样例占多大的比重。而在每次训练过程中，混合的数据集中的每个类别也会包含至少是样例数据集中那么多的数据。无论原始数据多么不平衡，至少样例集合是平衡的。

当样例集合E计算完毕后，通过

$$r^i = \overline{\hat{e}^i} \tag{3.7}$$

即可计算出类均值集合R。虽然这一部分计算不会对训练过程起作用，但这一部分的计算依然在服务器上完成。因为在推理过程时不需要用到样例集合E。而相较与样例集合E来说均值集合R的数据量较小。提前在服务器上完成这一运算可以减少数据传输量。

这一段将阐述训练分类网络ϕ时使用的损失函数。损失函数L由三部分组成，分别是分类损失$\mathrm{L_C}$，蒸馏损失$\mathrm{L_D}$，以及正则化损失$\mathrm{L_W}$。即

$$\mathrm{L} = \mathrm{L_C} + \mathrm{L_D} + \mathrm{L_W} \tag{3.8}$$

分类网络ϕ的模型结构已介绍过，是多层全连接层。训练分类网络ϕ所使用的数据集上文也已经讲过，是新数据集混合样例数据集F。对于新数据集中的数据以及样例数据集中的数据，训练时使用不同的损失函数。对于新数据集来说，训练的目标是使新数据集中的数据被分到正确的类别中，对应的损失项是分类损失$\mathrm{L_C}$，它被定义为

$$\mathrm{L_C} = -\sum_{f_j^i \in F} \log \phi^i(f_j^i) \tag{3.9}$$

式中的ϕ^i代表了分类网络ϕ在f_j^i所属的类别i上的概率输出。ϕ使用 softmax 作为输出层的激活函数，而上式就是普通的 softmax 分类损失。对于样例数据集来说，训练的目标是保持数据在分类网络ϕ上的输出不变，对应的损失项是蒸馏损失$\mathrm{L_D}$，它被

定义为

$$\mathrm{L_D} = -\sum_{f\in E}\sum_{k=1}^{n}\tilde{\phi}^k(f)\log\phi^k(f)+(1-\tilde{\phi}^k(f))\log(1-\phi^k(f)) \tag{3.10}$$

式中的$\tilde{\phi}$是分类网络ϕ在训练之前的拷贝。蒸馏损失$\mathrm{L_D}$其实是分类网络ϕ训练之前与训练之后对于样例数据集输出的交叉熵。设置蒸馏损失$\mathrm{L_D}$能固定分类网络ϕ对于样例数据集的输出分布。此外，对于分类网络ϕ的所有权重，模型设置了L^2正则化损失$\mathrm{L_W}$以避免网络过拟合。

最后总结一下整个训练的过程。当终端设备接收到已标注的图片时，终端设备会提取图片的特征，并将特征传输到服务器端。这些特征在服务器端累积到一定的数量后，会与样例数据混合，作为训练数据来训练分类网络。训练过程完成后，样例选取程序会从这个混合数据集中抽取出一批数据作为新的样例数据。当位于网络设备上的推理程序需要更新时，服务器端会将转换网络与由最新样例数据计算出的各类均值发送到网络设备中。

3.2.3 实时推理

这一小节将介绍推理过程。当终端设备接收到未标注的图片x时，终端设备会提取图片的特征f，并将特征f传输到网络设备。特征经过位于网络设备上的转换网络φ后变为特征$\hat{f}$。再通过最近类均值分类即可得到相应的类别y，最近类均值分类的公式为

$$y = \min_{y=1}^{n} \| \hat{f} - r^y \| \tag{3.11}$$

该公式的含义为选取距离$\hat{f}$最近的一个类均值，以其类别归属作为结果。得到的结果y将被传输给终端设备进行展示。

3.3 实验结果及分析

本节展示了三个实验。这三个实验分别针对了终端设备，网络设备，以及服务器。

特征提取模块被部署在终端设备上，本节测量了当使用树莓派作为终端设备时能够达到的特征提取帧率以及此时的带宽占用情况。实时推理模块被部署在网络设备上，本节测量了以树莓派作为网络设备时的负载能力。增量训练部分被部署在服务器上，本节测量了增量训练方法所能够达到的准确率。

3.3.1 实验环境

实验所使用的硬件设备包含了一台树莓派以及一台服务器。树莓派的型号是 Raspberry Pi 3 Model B，搭载着 Quad Core 1.2GHz Broadcom BCM2837 64bit CPU 以及 1GB RAM。实验中终端设备与网络设备的载体都是这块树莓派，它承担了对于特征提取程序以及实时推理程序的测试工作。服务器上搭载着 Intel(R) Core(TM) i7-6700K CPU，GeForce GTX 1080 GPU，以及 32GB RAM。这台服务器承担了对于增量训练过程的测试工作。在本节的实验环境下，设备之间通过 WIFI 网络进行连接。

实验所采用的具体模型结构为 MobileNet[24]加上两层全连接层。其中以 MobileNet 作为特征提取网络，两层全连接层合在一起作为分类网络，两层全连接层中与特征提取网络相连的那一层作为转换网络。MobileNet 是 Google 为了移动端图像识别应用而提出的网络结构，它使用了深度方向分离的技术以减少卷积层参数个数。此外，MobileNet 还提供了了两个超参数，通过调节这两个超参数的取值可以在精度与资源消耗之间做权衡。Google 提供了在 ILSVRC-2012-CLS[53]数据集上经过预训练的 MobileNet 模型供应用开发者使用。本节选取了 mobilenet_v1_0.50_224 这一版本，它具有 1.34M 的参数，推理过程需要 150M 次乘积累加运算，在 ILSVRC-2012-CLS 上的 top-1 准确率为 63.3，top-5 准确率为 84.9。本节设置特征提取网络的输出维数即 p 的值为 1001。转换网络的输出维数即 q 的值为 512。

实验所使用的数据集为 Core50[54]。本系统的应用方向为识别出用户曾经教给系统的物体。根据这个特性，本节选取了 Core50 作为实验数据集。Core50 是一个为单物体持续学习而设计的图像识别数据集。这个数据集包含了 50 个不同的物体在 11 个不同的场景下拍摄的图片。每个物体在每个场景下都包含了一段帧率为 20 的持续时间为 15 秒的视频。为了增加输入图片之间的差别，本实验加大了图片之间的时间间隔，对每个物体在每个场景下的情况选取了 30 张图片。总共是分为 50 个类别的

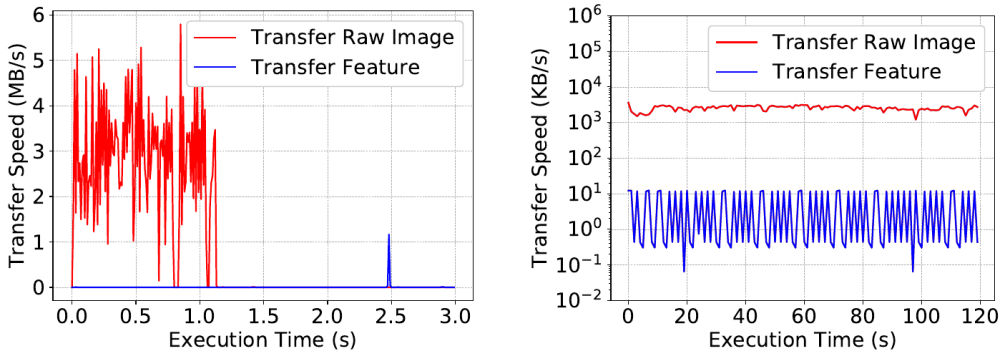
165,000 张图片。物体在单个场景下拍摄到的图片是有时间顺序的。本实验使用先拍摄到的 14,000 张图片为训练集，后拍摄到的 2,500 张图片作为测试集。

在网络设备以及服务器上都设有缓冲区。这两个部分是提供一对多服务的，在系统运行时，数据不一定会匀速的生成，缓冲区可以很好的应对这种情况。推理过程注重实时性，一旦缓冲区中有了数据，推理程序就会对数据进行处理。每次增量训练过程则需要一定数量的新数据，只有当缓冲区中的数据数量高于这个数字时才会进行训练操作。

3.3.2 特征提取操作表现

这一小节中将展示针对终端特征提取操作所做的实验。在终端设备上部署特征提取程序能带来降低数据传输，增加数据传输安全性，分担运算等好处。这个实验则专注于特征提取操作对终端设备性能的影响。实验测量了在执行特征提取操作时终端设备的执行速度以及网络资源占用情况。

图 3-3(a)展示了处理单张图片时的带宽占用情况。这张原始图像的传输数据量为 2.98 MB，提取出的特征的传输数据量为 10.3 KB。Transfer Feature 是先提取特征，然后将特征传输到网络设备时的带宽占用情况。作为对比实验，Transfer Raw Image 是直接传输原始图像时的带宽占用情况。在直接传输原始图像的情况下，终端设备在取得图像后立即开始了传输过程，整个过程在 1.2 秒内完成。而在先提取特征后传输的



(a) (b)

图3-3 特征提取过程在终端执行时的帧率以及带宽占用情况

情况下，终端设备先花费了大约 2.5 秒的时间进行特征提取操作，然后才将特征传输出去。可以看出，在这种情况下，直接传输原始图像的速度比传输特征的过程快出 2 倍左右。

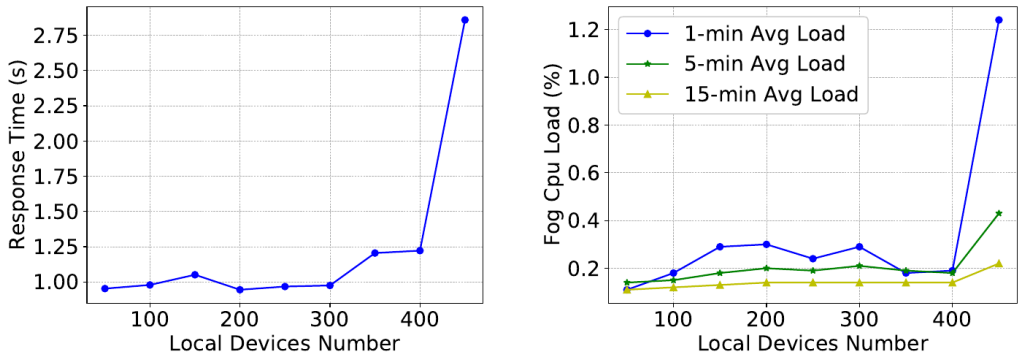
图 3-3(b)展示了处理连续图片时的带宽占用情况。整个实验持续了 120 秒，所有原始图像的平均传输数据量为 3.906 MB，而特征的平均传输数据量为 11.466 KB。Transfer Feature 与 Transfer Raw Image 的含义与图 3-3(a)中一致。传输原始图像时的的带宽占用为 2602.241 KB/s，帧率为 0.65。传输特征时的带宽占用为 31.341 KB/s，帧率为 0.51。结果显示，在这种情况下，传输特征的帧率略低，但却大大降低了带宽占用。

从以上的实验中可以看出两点：一是诸如树莓派这样的硬件设备可以负担起诸如 MobileNet 这样的特征提取网络，并达到秒级的图像处理速度。二是特征提取操作可以减少带宽占用。虽然原始图像尺寸，实验设备的带宽等因素都会对实验结果造成影响，但以上两点大致可以确定。

3.3.3 网络设备负载能力

这一小节展示了针对网络设备负载能力所进行的实验。在本章的框架中，网络设备往往要服务多个终端设备，且要处理高频率的推理请求。然而网络设备的运算能力却不如服务器，对于网络设备来说，它处理并发任务的能力值得验证。此次模拟实验使用树莓派作为网络设备，在服务器上开了并行的线程模拟终端设备向网络设备发送请求，以此来检验网络设备的负载能力。在服务器上，本实验从 0 开始，每 30 秒增加 50 个模拟终端设备的线程，整个实验持续了 5 分钟，线程的数量达到了 500。每个线程进行请求的时间是随机的，平均每 6 分钟发起一次推理请求，平均每 30 分钟发起一次上传已标注数据请求，平均每 24 小时发起一次模型更新请求。

图 3-4 展示了实验结果。其中图 3-4(a)展示了推理运算响应延时与终端设备数量



(a) (b)

图3-4 网络设备的负载能力

的关系，而图 3-4(b)展示了网络设备的 CPU 负载率与终端设备数量的关系.从图 3-4(a)与 3-4(b)中可以看出中可以看出，当设备数量超过 400 时，网络设备的响应时间以及 CPU 占用显著升高，此时已经出现了数据拥塞现象。

3.3.4 增量学习的准确度

如图 3-5 与图 3-6 所示，这一小节展示了增量学习方法的准确率以及资源占用情况。实验使用的数据集 Core50 在 3.3.1 节已经介绍过。实验使用随机梯度下降算法，设置训练时每批数据的数量为 128 个，每批迭代训练 60 次。除了本文中介绍的增量学习方法 ILIoT(Incremental Learning for IoT Systems)外，实验中还设计了其

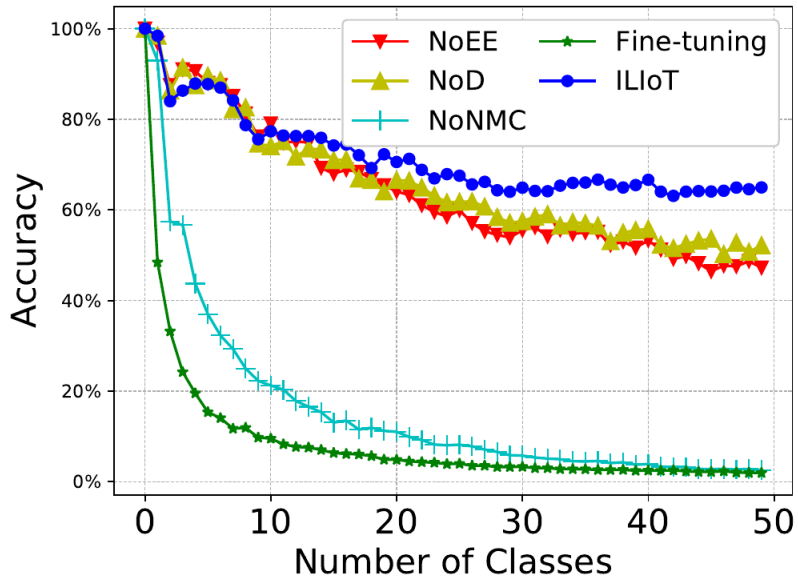
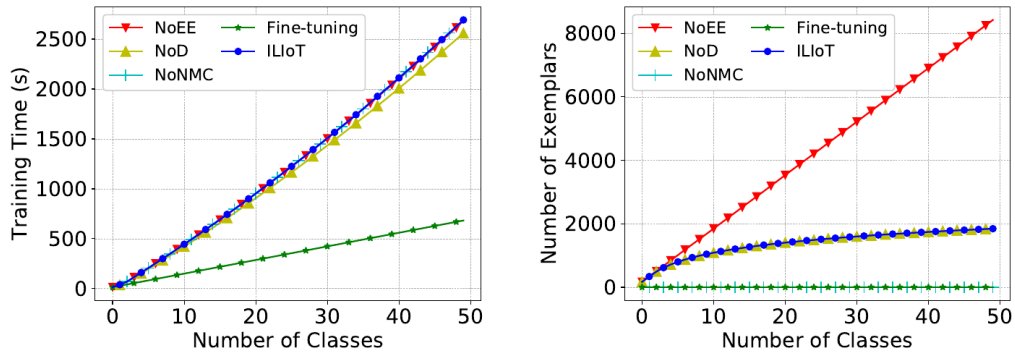


图3-5 增量学习的准确率

他的一些对比实验。NoEE(No Exemplars Extraction)指的是省去特征提取过程的情况，使用这种方法时，增量训练程序会保留所有的历史数据作为样例数据。NoD(No Distillation)指的是没有蒸馏损失的情况，使用这种方法时，增量训练程序将样例数据视为新数据一样处理，在训练时使用分类损失来保证这些数据输出正确的分类结果，而不是使用蒸馏损失固定其输出。NoNMC(No Nearest-mean-of-exemplars Classification)指的是不使用最近类均值分类的情况，此时使用分类网络输出的类别作为分类结果。Fine-tuning 指的是不使用任何增量学习技术的情况，此时每次直接在新的数据集上训练分类网络。

图 3-5 与图 3-6 展示了将 Core50 数据集中的数据按类别逐个教给模型时的分类



(a) (b)

图3-6 增量学习的时间消耗以及空间占用

准确度以及时间，空间资源的占用与已习得类别数的关系。如图 3-5 所示，当类别数量大于 10 时，本文的方法所得到的准确率超过了其他方法，最终能达到 0.650 的分类准确率。图 3-6(a)展示了总训练时间与已习得类别数的关系。从图中可以看出所有的实验方法消耗的总时间都大致呈线性增长，即每次训练一个新类别的时间都大致相同。这是因为实验每次输出的数据总量，每批大小，迭代次数等会大幅影响训练时长的参数是保持不变的。除去分类准确率大幅下降的 Fine-tuning 方法外，其余方法的训练时间消耗都大致相同。图 3-6(b)展示了总样例数据数量与已习得类别数的关系。可以看到 NoEE 方法的样例数量是随着已习得类别数线性增长的，而 Fine-tuning 方法不保留任何样例数据。其余三种方法保留的样例数量收敛于用户给出的样例集合容量。可以看出，随着类别数量的升高，本文提出的方法能取得最高的分类准确率，消耗的时间，空间资源也在有限的范围内。

3.4 本章小结

针对物体认知系统的图像处理能力，本章提出了一种由特征提取，增量训练，以及实时推理三个模块组成的可增量的图像识别框架。部署在该框架下的图像识别应用拥有增量学习的能力。此外，对这三个模块的合理分配可以达到降低互联网应用后期维护成本，减少网络传输消耗，加快响应速度，提高安全性等目的。本章在如今常见的单片机与服务器上设计了一系列的实验。实验结果表明这些设备的处理能力足以支撑起本章提出的框架。

第四章 多层神经网络推理过程切分算法

从以云为中心的集中模式到边缘计算等稀疏分布模式，物联网应用的运算分配一直是研究的热点。优秀的运算分布方法可以充分利用系统资源，提高响应速度。本章针对物体认知系统的运算分配问题，以提高系统响应速度为目标，以如今物联网中的多层架构为背景，设计出了一种多层神经网络推理过程切分算法，为智能物联网运算分布问题提供了新的解决思路。

4.1 系统模型与优化目标

多层物联网架构以及深度学习都是较为新兴的技术，对于深度学习的推理过程在多层物联网架构中的运算分配问题尚没有成熟的模型被建立起来。这一节将定义系统模型，以及优化目标。

4.1.1 系统模型

图 4-1 展示了本章建立的推理过程运算分配模型。本章要将多层神经网络的运算

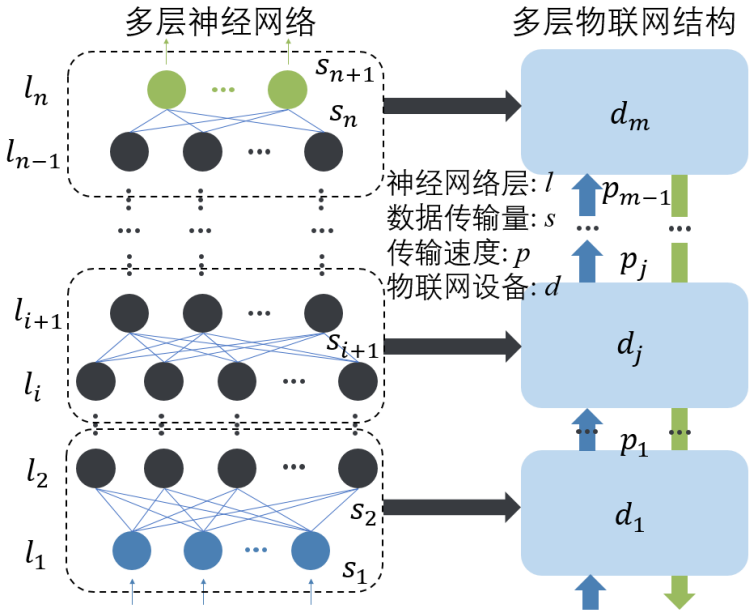


图4-1 推理过程运算分配模型

切分成若干块，分别分配给各层物联网设备。在本章建立的模型中，数据从第一层，也就是最底层的物联网设备中产生，这一层的物联网设备对应于现实中的智能摄像头，

手机等带有一定处理能力的物联网终端设备。每一层设备都可以选择负担零到多层的神经网络推理过程的运算，然后将运算得到的结果传输给上一层设备，例如终端设备将结果传输给网络设备，网络设备将结果传输给服务器。直到整个神经网络的推理过程执行完毕，最终结果将被逐级向下传播到第一层，然后反馈给用户。用形式化的语言来表示则为，要将 n 层的神经网络推理运算 $L=\{l_0,l_1,...,l_n\}$ 分配给 m 层的物联网设备 $D=\{d_1,d_2,...,d_m\}$。本章使用 $A=\{a_1,a_2,...,a_n\}$ 来表示一种分配策略， $a_i \in 1,...m$ 并且 $a_{i+1}>=a_i$。其中 a_i 表示第 i 层的神经网络推理运算 l_i 被分配给了设备 d_{a_i}。

本章设定 $a_{i+1}>=a_i$，接下来将证明这样的设定不会使模型错过优秀的分配策略。在本章的场景中，原始数据从第一层设备中产生，然而一些高层设备拥有更强大的运算能力。将运算分配给更高层意味着牺牲传输数据所消耗的时间来节省运算所消耗的时间。可以证明，在这样的模型中，将运算传输给更低层的设备不会带来任何的好处，只能白白消耗传输所需的时间。证明过程如下：

需要证明将数据向下传播不会带来好处。将数据向下传播则等价于这样一种场景，运算 x 被分配给了 d_i，然后有运算 y 被分配给了 d_{i+j}，其中 $i,j>=1$，然后又有运算 z 被分配给了 d_i。数据向下传播会带来好处则等价于，存在着这样一种方案，它满足数据向下传播了的要求，并且比所有不向下传播的方案要好。在此选择两种不向下传播的方案，将这个问题转化为证明该方案

（1）消耗的时间比运算 xyz 全被分配在 d_i 的情况要短。

（2）消耗的时间比运算 x 被分配在了 d_i，而运算 yz 被分配在 d_{i+j} 的情况要短。使用反证法，假设有这样的情况成立，首先要满足（1），由于在此场景下多出了将 x 的运算结果从 d_i 传输到 d_{i+j} 的时间以及将 y 的运算结果从 d_{i+j} 传输到 d_i 的时间，因此要满足 y 在 d_{i+j} 上消耗的时间比在 d_i 的时间要短，因为传输消耗的时间大于 0，所以这边的时间是严格的要短，也就是说

（3） d_{i+j} 运行速度要比 d_i 快。

同时还要满足条件（2），也就是说 xz 在 d_i，y 在 d_{i+j} 的情况要比 x 在 d_i，yz 被分配在 d_{i+j} 的情况要快，前者与后者相比，多出了将 y 的运算结果从 d_{i+j} 传输到 d_i 的时间，所以 z 在 d_i 上的运算速度一定要比 z 在 d_{i+j} 上的运算速度快，也就是说

（4）d_i 运行速度比 d_{i+j} 快。

（3）与（4）矛盾，即（1）与（2）矛盾，不能同时成立，即证明假设的方案不成立，即证明不存在数据向下传播会带来好处的方案。用形式化的语言来表达则为，l_i 之后的运算只能被分配到 d_{a_i} 以及之后的设备中，再简化一点则为 $a_{i+1} >= a_i$。

4.1.2 优化目标

本章的模型以系统响应时间T 作为评价标准。响应时间T 有两个组成部分，分别是传输时间 T_{trans} 以及运算时间 T_{calcu}。现有的研究通过建立预测模型来预测各层神经网络在各设备上的运算时间[15][55]。本章的方法是为多层物联网系统设计的，有很多不可预料的因素会影响设备的运算能力以及传输带宽。例如，就运行环境而言，在很多情况下存在着多个低层设备共同连接到一个高层设备的情况，这种共享会影响到高层设备的运算速度以及传输速度。由于低层设备的数量乃至于运行的频繁程度都是不确定的，甚至是会随时间改变的，这样的影响几乎没有办法被预测。因此本章没有建立模型来预测各层网络在各层设备上的运算时间，而是直接测量得到了它们。本章用 $C=\{c_1^1,...,c_i^j,...,c_n^m\}$ 来表示各层网络在各层设备上的运算时间，其中 c_i^j 代表 l_i 在 d_j 上运行所需要的时间。为了计算传输时间，需要测得各层网络之间的数据传输量 $S=\{s_1,s_2,...,s_{n+1}\}$，其中 $s_1,...,s_n$ 分别是网络 $l_1,...,l_n$ 的输入数据量，而 s_{n+1} 是 l_n 的数据输出量。此外还需测量得到传输速度 $P=\{p_1,p_2,...,p_{m-1}\}$，其中 p_j 代表设备 d_j 到设备 d_{j+1} 之间的数据传输速度。至此，本章已经定义完了所有的变量，这些变量之间的对应关系见图 4-1，厘清这些细节对阅读接下来的公式有很大帮助。运算时间 T_{calcu} 被定义为

$$\mathrm{T}_{calcu}=\sum_{i=1}^{n} c_i^{a_i} \tag{4.1}$$

这个很好理解，就是各层神经网络在各自对应的设备上运算的时间之和。而传输时间则被定义为

$$\mathrm{T}_{trans}=\sum_{i=1}^{n}\sum_{j=a_{(i-1)}}^{(a_i)-1}(s_i+s_{n+1})/p_j \qquad (a_0=1) \tag{4.2}$$

第一重循环是对所有神经网络层的，第二重循环则是对两层神经网络间的所有物联网设备。当相邻的两层神经网络没有被分配到同一个物联网设备上时，$a_{(i-1)}<=(a_i)-1$ 这个条件成立，这第二重求和将计算 $d_{a_{i-1}}$ 到 d_{a_i} 之前所有的传输时间消耗。这个传输消耗包括了将 l_i 的输出从 $d_{a_{i-1}}$ 传输到 d_{a_i} 的时间，也包括了将整个网络的运行结果从 d_{a_i} 传输到 $d_{a_{i-1}}$ 的时间。如图 4-1 所示，公式中的 s_i 是 l_i 的输入数据量，而 s_{n+1} 是网络推理结果的数据量。公式中的 p_j 是 d_j 到 d_{j+1} 的数据传输速度，其中 $j\in a_{i-1},...,(a_i)-1$。而 $a_0=1$ 则代表数据产生于设备 d_1。最后，响应时间 T 被定义为

$$\mathrm{T}=\mathrm{T}_{calcu}+\mathrm{T}_{trans} \tag{4.3}$$

总的来说，通过对关键信息的提取，本节将神经网络的分配定义为了一个最优化问题：给定各层神经网络在各设备上的运算时间 C，各层神经网络的输入输出数据量 S，物联网设备间的数据传输速度 P，需要找到一种分配策略 A 来最小化响应时间 T。

4.2 网络切分算法

这一节将阐述为了解决在上一节所提出的最优化问题而提出的网络切分算法。

4.2.1 问题规模分析

为了对问题规模有直观的影响，在此先分析一下使用穷举法共有多少种可能性。问题等价于将 n 层神经网络塞进 m 层不同的设备，并且允许有空的设备，这是个很经典的组合数学问题。如果不允许有空的设备，就能将这个问题简化为将 n 层神经网络切分成 m 块，然后分别被塞进这 m 个设备中。这就等价于将 $m-1$ 块隔板塞进 $n-1$ 个缝隙中的方案数量。当给定网络层数 n 以及设备数量 m 时，可能的方案有 C_{n-1}^{n-1} 种，其中 C 是组合数。回到允许空设备的情况，为了使用刚刚得到的结论，可以将这个问题

转化为将$n+m$层网络切成m份，然后每一份包含的网络层数减1。套用之前的结论，可以得到可能的策略数为C_{m-1}^{n+m-1}。由此可以得出穷举法的时间复杂度为$O((n+m-1)!/(n!(m-1)!))$。这个数字会随着m和n的增长而迅速增加，因此需要一个复杂度更低的切分算法。

4.2.2 状态变量设计与计算

本章提出了一种动态规划算法来解决这个问题。本章设定了状态变量$H=\{h_1^1,...,h_i^j,...,h_{n+1}^m\}$，其中$h_i^j$代表了满足将网络层$l_i$的输入数据传输到设备$d_j$的条件时所能达到的最短响应时间。有一个例外，$h_{n+1}^j$代表了将整个网络的输出结果传输到设备$d_j$时所能达到的最短响应时间。值得注意的是，当计算$h_i^j$时，并不需要关注$l_{i-1}$或$l_i$是在哪一个设备中进行的运算，只要求$l_i$的输入数据已经被传输到了$d_j$。后文将详细阐述这样定义的原因。接下来将阐述$H$的计算方式。当$i=1$时，通过以下公式计算$h_1^j$。

$$h_1^j=\begin{cases}0 & j=1\\ h_1^{j-1}+(s_1+s_{n+1})/p_{j-1} & j>1\end{cases} \tag{4.4}$$

当$i=1$时，只需要考虑l_1的输入，也就是神经网络的原始输入被传输到网络各层时的情况。此时只有数据传输时间消耗，也就是将原始输入传输到网络各层，然后将结果传输回最低层的时间消耗。值得注意的是，这边的回传结果是整个网络的计算结果而不是已经进行的计算所得到的结果。因为最终回传的只有整个网络的计算结果，关注中间结果的回传没有任何意义。当$j=1$时，由于输入数据本来就在设备d_1中，没有必要进行任何数据传输，此时消耗的时间为0。当$j>1$时，响应时间h_1^j是d_1到d_j所需的数据传输时间，也就等于d_1到d_{j-1}所需的数据传输时间h_1^{j-1}，加上d_{j-1}到d_j的数据传输时间$(s_1+s_{n+1})/p_{j-1}$。至此，已经讨论完了$i=1$时的情况。当$i>1$时，通过以下公式计算h_i^j。

$$h_i^j = \begin{cases} h_{i-1}^1 + c_{i-1}^1 & j = 1 \\ \min(h_{i-1}^j + c_{i-1}^j, h_i^{j-1} + (s_i + s_{n+1}) / p_{j-1}) & j > 1 \end{cases} \tag{4.5}$$

当 $j=1$ 时，所有的计算都在设备 d_1 上执行，此时没有传输消耗。响应时间 h_i^j 是从 l_1 到 l_{i-1} 的网络层在设备 d_1 上执行所消耗的运算时间。也就是 l_1 到 l_{i-2} 消耗的运算时间 h_{i-1}^1，加上 l_{i-1} 在设备 d_1 上执行所消耗的运算时间 c_{i-1}^1。当在 d_1 上完成了 l_{i-1} 运算后，l_{i-1} 的运算结果，也就是 l_i 的输入就在 d_1 上了。这与 H 的定义一致。当 $i, j > 1$ 时，要到达状态 h_i^j 有且只有两条路。第一条路是通过 h_{i-1}^j，这意味着此时 l_{i-1} 所需的输入数据位于 d_j 上，此时需要消耗 c_{i-1}^j 的时间来在 d_j 上进行 l_i 的运算，然后 l_i 的输入便在 d_j 上了。第二条路是通过 h_i^{j-1}，这意味着 l_i 的输入已经被计算出来，并且位于设备 d_{j-1}，需要消耗 $(s_i + s_{n+1}) / p_{j-1}$ 的时间来在 d_{j-1} 与 d_j 之间进行数据传输。通过选择两条路中响应时间较短的一条，可以计算出 h_i^j。至此，已经完成了对状态变量 H 的计算。

4.2.3 获取最优运算分配策略

完成对状态变量 H 的计算后，便可以很方便的获得最优的运算分配策略。最短响应时间可以通过以下公式计算。

$$\mathrm{T} = \min_{j \in 1,\ldots,m} h_{n+1}^j \tag{4.6}$$

这个公式的含义为从完成整个网络计算的所有策略中选取响应时间最短的一种。可以

算法 4-1：取得分配策略

Input : 状态变量 $H = \{h_1^1, \ldots, h_i^j, \ldots, h_{n+1}^m\}$
Output: 分配策略 $A = \{a_1, a_2, \ldots a_n\}$
$a_n = \arg\min_{j \in 1,\ldots,m} h_{n+1}^j$
$t = a_n$
for $i \in n-1, \ldots, 1$ **do**
  while $h_{i+1}^t \neq h_i^t + c_i^t$ **do**
    $t = t - 1$
  $a_i = t$
return A

通过算法 4-1 计算出分配策略 A。本算法从神经网络的最顶层开始逐层向下寻找每一层运算所分配到的设备。a_n 是分配策略所使用的最后一个设备。t 代表当前设备，因此最开始 t 等于 a_n。若 $h_{i+1}^t \neq h_i^t + c_i^t$，则 l_i 的运算不是在 d_t 上进行的。此时将逐级向下直到找到进行 l_i 的运算的设备。重复这一过程就能获得完整的分配策略 A。至此整个网络切分算法已经阐述完毕。

4.3.4 算法分析

接下来计算一下算法的时间复杂度。H 包含了 $m(n+1)$ 个元素，每个元素可以在 $O(1)$ 的时间内被计算出来，因此计算 H 的时间复杂度为 $O(mn)$。计算完 H 后，需要通过算法 4-1 得到分配策略 A。找到 a_n 的时间复杂度是 $O(m)$。此后，为了找到从 h_i^1 到 $h_{n+1}^{a_n}$ 的一条路径，需要经过 $n+m$ 个中间状态，经过每个中间状态的时间复杂度是 $O(1)$，因此算法 4-1 的时间复杂度是 $O(m+n)$。而完整算法的时间复杂度是上述两个步骤的和，为 $O(mn)$。

接下来将论述为什么算法可以找到最优分配策略。在本章的模型中，神经网络层的运算是连续且单调的。此外，在模型中数据在设备间的传输是连续且单调的。也就是说，不存在能越过某个神经网络层的推理运算，也不存在能越过某个设备的数据传输，并且推理运算和数据传输都是不能回头的。这才有了之前所说的为了到达状态 h_i^j，必须要经过状态 h_i^{j-1} 或者状态 h_{i-1}^j。这样才能说该动态规划算法内包含了所有的可能性，才能说它可以找到最优分配策略。这也是本算法将 h_i^j 定义为满足将网络层 l_i 的输入数据传输到设备 d_j 的条件时所能达到的最短响应时间的原因。这个定义暗示了网络层 l_{i-1} 的运算已经被执行，并且指明了运算结果已经被传输到了设备 d_j 上。算法本章提出的算法和穷举法一样都能取得最优解，然而计算的时间复杂度则从 $O((n+m-1)!/(n!(m-1)!))$.降到了 $O(mn)$。接下来将通过实验展示算法的可行性以及优势。

4.3 实验结果及分析

本节设计了一个模拟实验来验证网络切分算法的表现。这个实验的主要目的是验证算法的正确性，以及展示在现实设备中应用该算法的可能性。实验用本章提出的算法在四层模拟物联网环境中分割了一个图像分类网络的推理过程。

4.3.1 实验环境

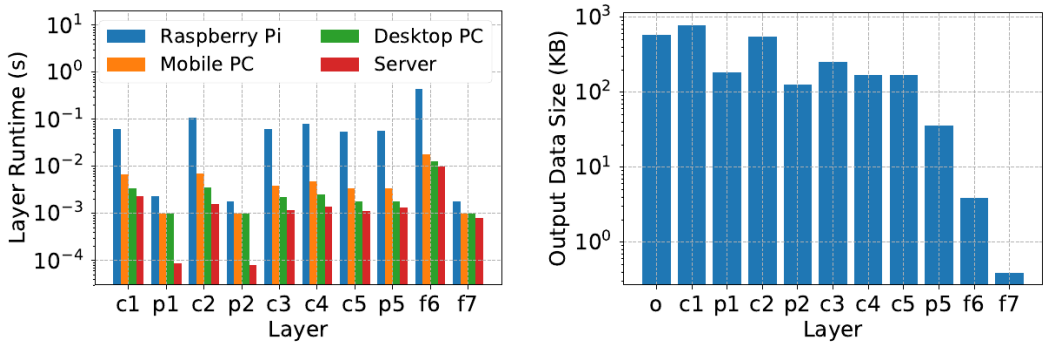
图像分类网络的结构仿照 AlexNet[35]的结构进行设计。该网络的输入数据是 224 乘 224 乘 3 的 RGB 图像。网络共包含十层，分别是 $L=\{c1,p1,c2,p2,c3,c4,c5,p5,f6,f7\}$。其中包含了 5 层卷积层 $\{c1,c2,c3,c4,c5\}$，3 层池化层 $\{p1,p2,p5\}$，以及 2 层全连接层 $\{f6,f7\}$。网络切分算法只会改变推理运算在整个系统中的执行速度，对于分类精度不会有任何影响。因此实验也不会关注分类精度。算法对于数据集以及神经网络结构没有过多的要求。只要网络结构是分层的，并且每层间传输的数据量可以被测量得到，就可以应用该网络切分算法。

本节搭建的模拟物联网环境包含 4 层设备。它们分别是 $D=\{RaspberryPi,MobilePC,DesktopPC,Server\}$。$RaspberryPi$ 是一块 Raspberry Pi 3 Model B 模组，搭载着 Quad Core 1.2GHz Broadcom BCM2837 64bit CPU 以及 1GB RAM。$MobilePC$ 是一台笔记本电脑，搭载着 Intel(R) Core(TM) i5-4210H CPU 以及 12GB RAM。$DesktopPC$ 是一台台式机，搭载着 Core(TM) i7-6700 CPU 以及 16GB RAM。$Server$ 是一台服务器，搭载着 Intel(R) Core(TM) i7-6700K CPU，GeForce GTX 1080 GPU，以及 32GB RAM。

4.3.2 算法输入

经过测量得到了算法所需的输入数据中各层网络在各层设备上的运算时间 C，以及各层网络之间的数据传输量 S。图 4-2 中展示了测量得到的数据。图 4-2(a)展示了各层网络在各层设备上的运算时间。可以看出设备的运算能力呈逐层上升的趋势。此外，每一项数据跟所处理的神经网络层，以及进行运算的设备都有关。例如设备 $Server$ 处理 $p1$ 和 $p2$ 时就特别快。图 4-2(b)展示了各层网络的输出数据量，其中横轴上的 o

代表了原始数据的大小。可以看出数据量随着网络层数的上升呈现出递减的趋势。

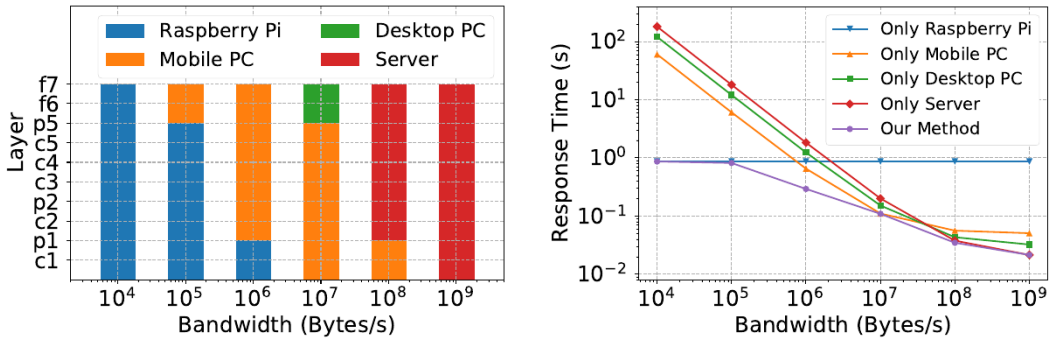


(a)　　(b)

图4-2 在实验环境中测得的算法参数

4.3.3 实验结果

本节将网络传输速度 P 作为自定义变量，调用了算法进行了模拟实验。实验结果如图 4-3 所示。模拟实验中设定各级设备间的数据传输速度都相同，即设定 $p_i = p_j$，



(a)　　(b)

图4-3 模拟实验结果

其中 $i, j \in 1,2,3$。传输速度的取值根据当前各种通信方式的速度进行选取。窄带物联网的最高传输速度为 8 KB/s[56]，而 5G 网络的峰值速度高达 20 Gb/s[57]，即 2.5 GB/s。据此，本节设定了模拟实验的网络传输速度取值范围为从 10 KB/s，即 10^4 B/s 到 1 GB/s，即 10^9 B/s。图 4-3(a)展示了算法输出的网络切分方案随传输带宽的变化。其中横坐标指明了传输带宽，柱状图的颜色代表了纵坐标所对应的各神经网络被分配到的设备。从图 4-3(a)中可以看出，随着传输带宽的升高，算法倾向于将运算分配到更高层的设备。这样的分配方式也很符合直觉，更高层的设备拥有更强大的运算能力，可以更快

的完成运算。而随着传输速度的升高，将数据向上传输的代价则相应降低。此外，从图 4-3(a)中可以看出，算法倾向于在网络层 $p1$ 与 $p5$ 之后做分割。这是因为 $p1$ 与 $p5$ 的输出数据量相对较少，而且 $p1$ 与 $p5$ 之后的网络层的运算量相对较大。这些特点可以从图 4-2 中看出。图 4-3(b)展示了响应时间随着网络传输带宽改变而发生的改变。图 4-3(b)画出了将整个神经网络分别部署在单独的设备时的效果作为对比试验。从图中可以看出，在任何时候网络切分算法都能取得最好的结果。

4.4 本章小结

本章针对多层神经网络推理运算在多层物联网设备上的运算分配问题。与现有的神经网络推理过程切分算法[15][55]相比，本章首次将这个问题扩展到了多层物联网设备的领域。本章抽象了该运算分配问题，提取出运算时间，数据传输量，传输速度等各项关键数据，以最小化系统响应时间为评价标准，建立了系统模型。并基于该系统模型，提出了能在较低的时间复杂度内计算出最优神经网络切分策略的算法。本章用实验验证了该系统模型以及切分算法在当前的实际环境中的可行性以及有效性。

第五章 物体认知系统设计及实现

传统的图像识别应用往往针对某一领域单独进行设计，其适用范围较窄，且适应环境变化的能力较弱。本章基于可增量图像识别算法框架设计并实现了一套物体认知系统。本系统应用了基于深度学习的图像识别算法，其最大的特点是能在部署后持续的学习用户所教授的知识。这是一套高自由度，广适应性的系统，将应用的功能设计以及细化过程交由用户来实现，从而以极低的代价为用户提供个性化的解决方案。

5.1 物体认知系统整体框架设计

本章所设计的物体认知系统所包含的硬件设备包括一台手机，一台华为云服务器，以及一台电脑。手机作为移动端，主要负责图像的采集、特征提取操作、与网络设备的通信、以及与用户的交互等工作。华为云服务器作为网络设备，它具有固定的 IP 地址，是本系统中所有网络请求的服务端，主要起着数据转发的功能。电脑作为服务器，承担了增量训练以及实时推理的工作。

图 5-1 展示了本章介绍的物体认知系统的整体框架以及推理与训练流程。在这个

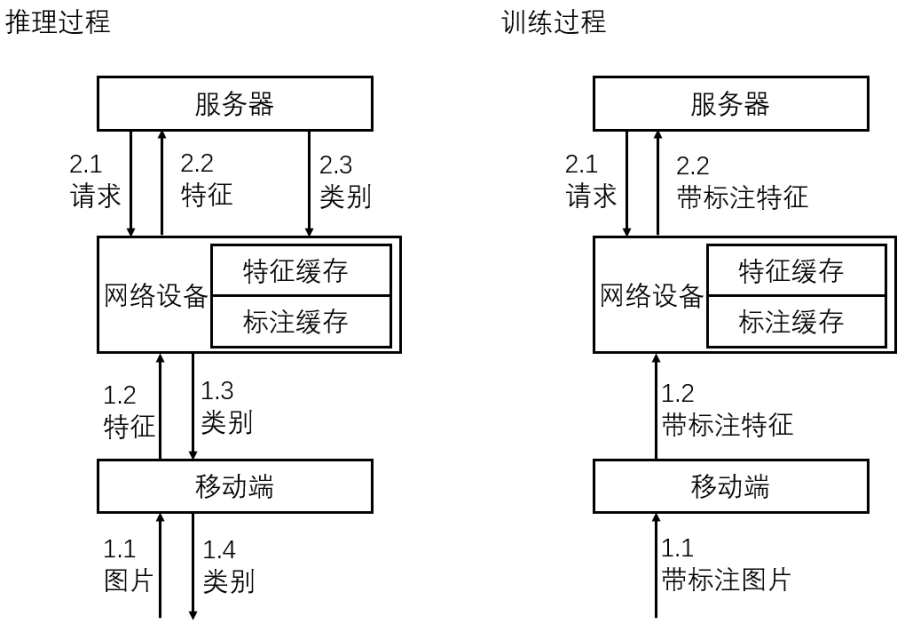


图5-1 物体认知系统整体框架以及推理与训练流程

框架下，移动端和网络设备之间的交互与服务器和网络设备之间的交互是不同步的。移动端和服务器都只对网络设备内的缓存数据进行操作。

在执行推理过程时，移动端与网络设备之间的数据传输流程为：1.1 移动端从摄像头采集图像，并进行特征提取操作。1.2 移动端将提取出的特征传输至网络设备，网络设备接收后将特征存储在特征缓存中。1.3 网络设备从标注缓存中读出类别信息，然后将类别信息传输至移动端。1.4 移动端将类别信息展示给用户。

执行推理过程时，服务器与网络设备之间的数据传输流程为：2.1 服务器向网络设备出特征下载请求。2.2 网络设备响应请求，将特征缓存中的特征传输给服务器。2.3 服务器进行推理操作，并将结果出输给网络设备，网络设备接收后将结果存储在标注缓存中。

在执行训练过程时，移动端与网络设备之间的数据传输流程为：1.1 移动端从摄像头采集图像，进行特征提取操作，并读取用户提供的标注。1.2 移动端将带标注的特征传输至网络设备，网络设备接收后将该带标注的特征存储在特征缓存中。

执行训练过程时，服务器与网络设备之间的数据传输流程为：2.1 服务器向网络设备出特征下载请求。2.2 网络设备响应请求，将特征缓存中带标注的特征传输给服务器。

5.2 物体认知系统各模块实现方案

本节提供了物体认知系统的的详细实现方式。本节分为三个小节，分别对移动端，网络设备端，以及服务器端的程序设计进行阐述。

5.2.1 移动端程序设计

移动端程序是一个安卓应用，开发环境为 Android Studio，开发过程中参考了 Google Developers Codelabs 下的项目。移动端程序包括了主程序循环，以及特征提取，网络传输两个模块。主程序循环负责响应用户操作、调取并显示摄像头预览图、显示程序运行结果等任务。特征提取模块负责提取图像的特征，网络传输模块负责与网络设备进行通信。

移动端应用的界面如图 5-2 所示。移动端应用可以工作在推理模式与训练模式两种模式下。运行在推理模式下时，应用持续的进行拍摄图片，提取特征上传至网络设备，显示网络设备返回结果的循环。运行在训练模式下时，应用持续的进行拍摄图片并提取特征，读取用户标注，将带标注的特征传送至网络设备，显示网络设备返回的

结果即上传的标注的循环。特征提取部分使用了 Tensorflow Lite[51]框架，使用在 1000 个类别的 ImageNet[53]数据集上预训练过的标准 MobileNet[24]作为特征提取模型。MobileNet 本身就是一个图像分类模型，应用会将 MobileNet 的分类结果显示在界面上，这一过程是离线的。网络传输部分调用了网络请求开源框架 OkHttp，使用 Post 请求与网络设备上运行的 WebService 服务器通信。上传数据时的帧结构在训练模式下为经过 Base64 编码的特征数组+分隔符+标注，在推理模式下为经过 Base64 编码的特征数组+分隔符+推理模式标识。

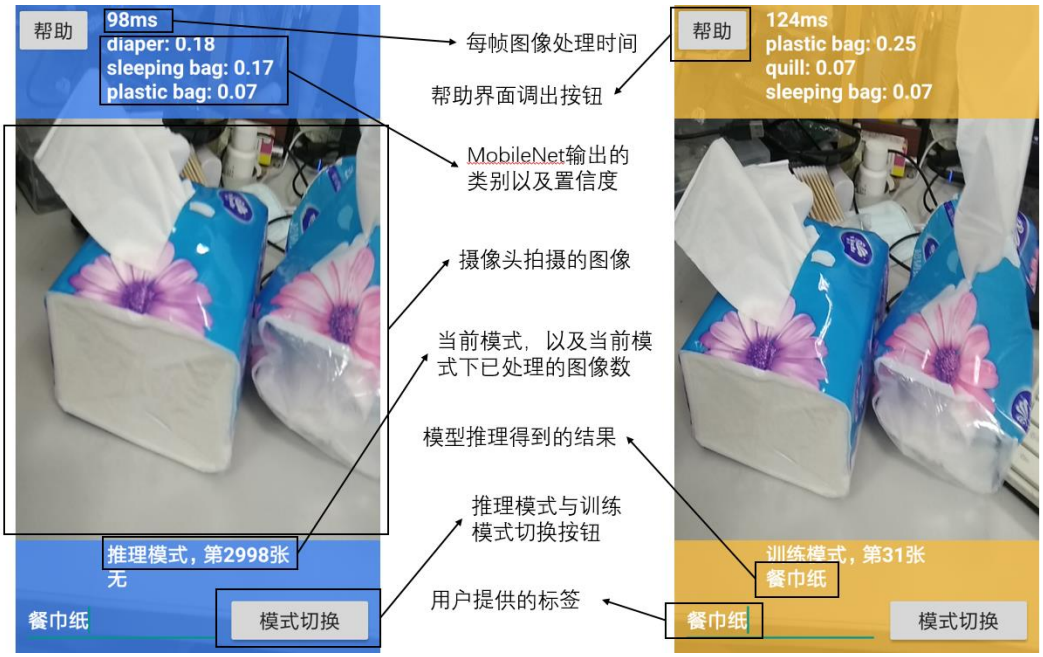


图5-2 移动端界面设计

5.2.2 网络设备端程序设计

网络设备具有固定的 IP 地址，是系统中所有网络访问的服务端。在网络设备上运行着一个通过 Microsoft Visual Studio 开发的 WebService 服务。如图 5-1 所示，网络设备对外提供三个接口函数，分别为上传特征接口，下载特征接口，以及上传标注接口。其中上传特征接口输入参数为移动端上传的特征帧，函数被调用后会把特征帧存入网络设备上的特征缓存中，函数返回为从标注缓存中读取的标注。下载特征接口没有输入参数，函数被调用后会返回从特征缓存中读取的特征。上传标注接口输入参数为字符串格式的标注，函数被调用后会将标注存入标注缓存。特征缓存与标注缓存分别是存储在网络设备上的两个文本文档。

5.2.3 服务器端程序设计

服务器端程序负责增量训练以及实时推理的运算。服务器端共包含三个程序，分别是数据传输程序，训练程序，以及推理程序。图 5-3 展示了服务器端推理与训练流程。在这个框架下，数据传输程序，训练程序，以及推理程序之间的交互是不同步的，它们都只对服务器内的缓存数据进行操作。

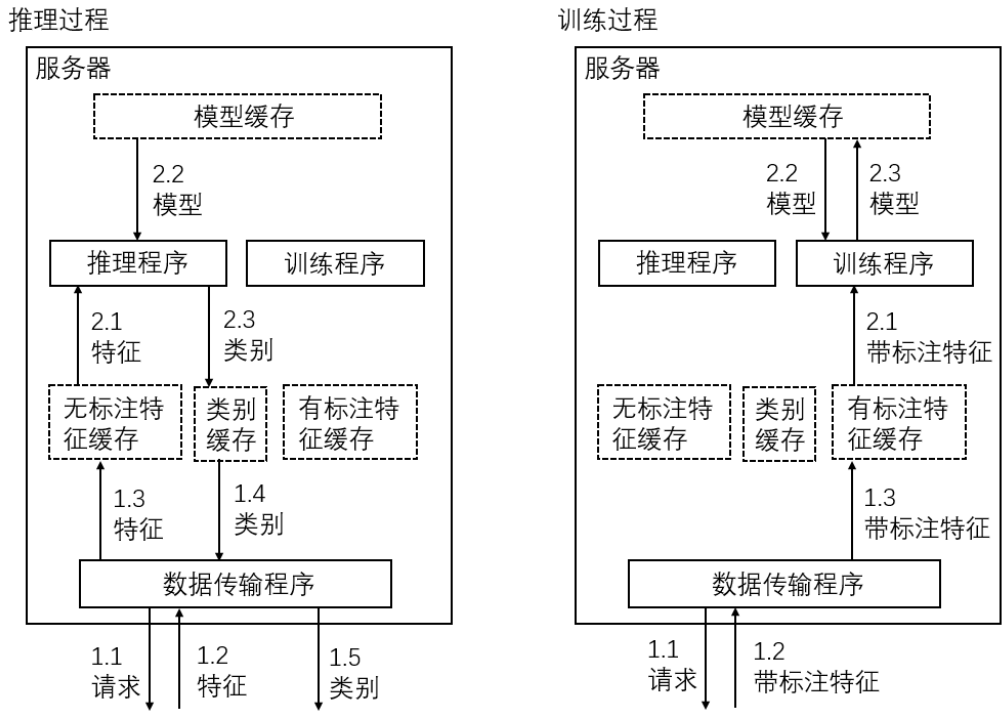


图5-3 服务器端推理与训练流程

在执行推理过程时，数据传输程序所执行的流程为：1.1 数据传输程序向网络设备发起特征下载请求。1.2 网络设备响应请求，将特征缓存中的特征传输给数据传输程序。1.3 数据传输程序比对新下载特征与上一帧特征是否一致，若新下载的特征与上一帧特征不一致，则将新一帧特征解帧，并将无标注特征保存至无标注特征缓存。1.4 数据传输程序从类别缓存中读取类别信息。1.5 数据传输程序通过标注上传请求将类别信息发送给网络设备。

在执行推理过程时，推理程序所执行的流程为：2.1 当推理程序检测到无标注特征缓存中有数据时，推理程序读取出无标注特征缓存中的所有数据，并清空无标注特征缓存。2.2 推理程序从模型缓存中读取推理程序所需要的模型部分，并完成推理过程。2.3 推理程序将所有特征的分类结果中占比最大的类别写入类别缓存中。

在执行训练过程时，数据传输程序所执行的流程为：1.1 数据传输程序向网络设备发起特征下载请求。1.2 网络设备响应请求，将特征缓存中的特征传输给数据传输程序。1.3 数据传输程序比对新下载特征与上一帧特征是否一致，若新下载的特征与上一帧特征不一致，则将新一帧特征解帧，并将带标注的特征保存至有标注特征缓存。

在执行训练过程时，训练程序所执行的流程为：2.1 当训练程序检测到有标注特征缓存中的数据数量大于训练所需最小数据量时，训练程序读取出有标注特征缓存中的所有数据，并清空有标注特征缓存。2.2 训练程序从模型缓存中读取训练程序所需要的模型部分，并完成训练过程。2.3 训练程序使用新训练出的模型更新模型缓存。

接下来将阐述一些实现的细节。数据传输程序是一个通过 Microsoft Visual Studio 开发的 C#应用程序。训练程序与推理程序分别实现了 3.2.2 中的增量训练方法与 3.2.3 中的实时推理方法。这两个程序以 python 语言编写，调用了 TensorFlow 工具包。分类网络由两层全连接层组成，分类网络输入参数个数即 3.2.1 节所介绍的参数 p 的值为 1001，两层网络中间参数个数即 3.2.2 节所介绍的参数 q 的值为 512，输出参数个数即最大可分类别数为 100。样例集合容量 v 为 500。训练所需最小数据量被设置为 10。训练会进行 60 个周期，每批训练数据大小为 200。服务器端的所有缓存都通过文本文件的方式实现。

5.3 系统测试

本节中展示了该物体认知系统的实际运行效果。本节分为三小节，第一小节简要介绍了实验环境，第二小节呈现了系统的运行流程，第三小节呈现了为展示系统的实用性而设计的不同的分类任务。

5.3.1 实验环境

实验所使用的硬件设备一台手机，一台华为云服务器，以及一台电脑。手机的型号是 Redemi Note 4X，拥有一颗高通骁龙 625 处理器以及 3.00GB 的内存，运行在 MIUI 10.2 操作系统下。华为云服务器拥有一颗 Intel(R) Xeon(R) Platinum 8163 CPU 以及 4.00GB 的内存，运行在 Windows Server 2012 R2 操作系统下。电脑拥有一颗 Intel(R) Core(TM) i7-7700 CPU，一颗 NVIDIA GeForce GT 730 的 GPU，以及 16.0GB

的内存，运行在 Windows 10 操作系统下。手机与电脑都通过 WIFI 上网。

5.3.2 运行流程

这一小节将展示使用物体认知系统的实际运行流程。用户可以自由的教给物体认知系统类别信息，这一教学的过程示例如图 5-4 所示。

图5-4 学习一个新类别的流程

如图 5-4(a)所示，在初始状态下，系统并不能认出拍摄到的餐巾纸，此时系统给出的标注为无。为了进行教学操作，用户首先应该做的是输入标注信息。如图 5-4(b)所示，在这个示例中，用户输入的标注信息为餐巾纸。然后用户需要按下模式切换按钮，将系统运行模式切换为训练模式。训练模式的界面如图 5-4(c)所示，此时用户需要拍摄该标注所对应的图片，在这个示例中，就是拍摄餐巾纸的图片。如 5.2.3 节中所介绍的，该物体认知系统训练所需最小数据量被设置为 10。为达到良好的识别效果，建议的教学一个新类别所需的最小数据量为 50，这一拍摄的过程大约能在 10 秒内完成。同时，服务器端执行一轮新的训练过程所消耗的时间在 5 秒以内。服务器端训练的过程与移动端的拍摄过程基本是同时进行的，当用户结束训练操作后，训练过程也将在若干秒后结束。可以说，用户不会感觉到服务器端训练所带来的延时。当教学完成后，用户需要再次按下模式切换按钮，将系统运行模式切换回推理模式。如图 5-4(d)所示，此时的物体认知系统已经可以识别出餐巾纸了。

5.3.3 系统实用性展示

本小节通过若干样例任务直观的展示该系统在多种场景下发挥的作用。在 3.3 节中针对可增量图像识别框架的运行速度，并发处理能力，分类准确率等性能指标做了

定量的实验。然而对于物体认知系统运行起来是什么效果，能够适应什么样的任务等问题，本文仍然没有给出直观的回答。本小节以若干样例任务为引，希望能定性的展示物体认知系统的若干功能，给读者留下一个直观的印象。

对于各类基本物体的识别是物体认知系统的基本功。图 5-5 展示了系统在训练模式下连续的学习了 8 个类别的物体后，进入推理模式后能取得的物体分类效果。

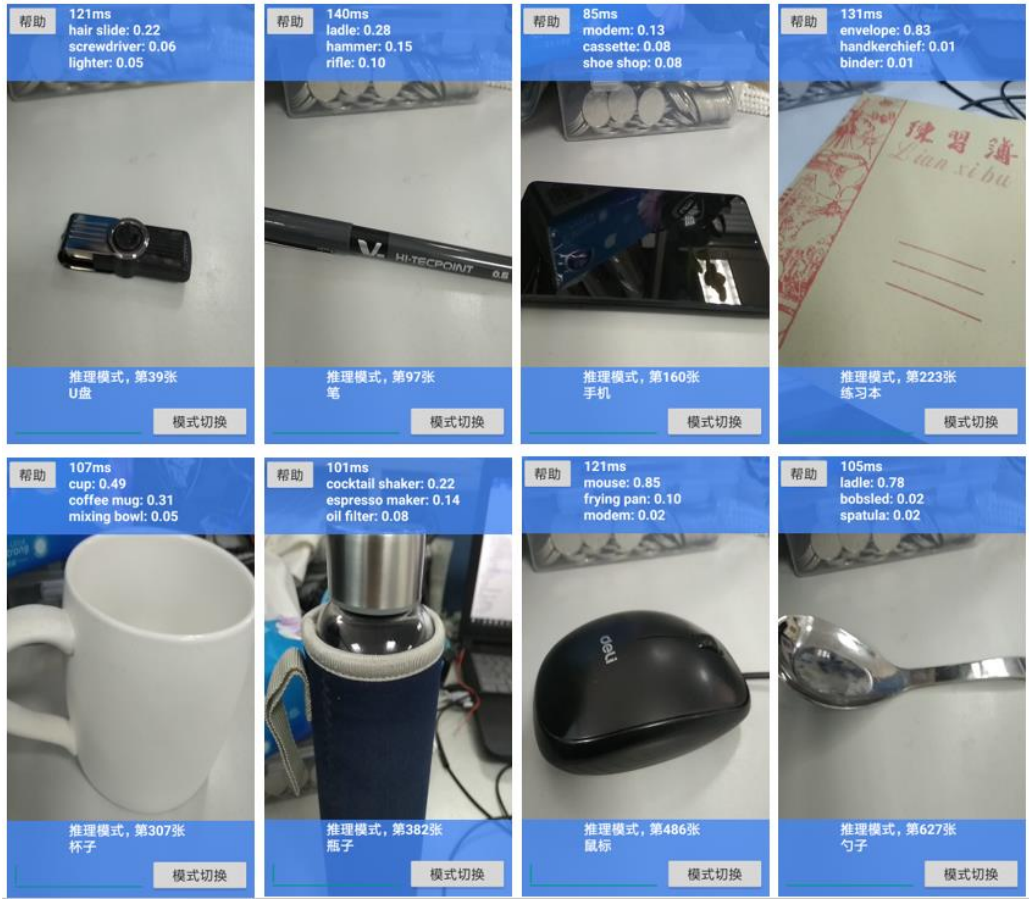


图5-5 系统能识别的各类物体

多角度多场景下的物体识别能力对于某些应用来说也非常重要。物体认知系统的终端程序位于手机上，这使得用户能够很方便的从各个角度采集物体的图像。此外，物体认知系统具备持续学习的能力，这使得用户可以随时补充某个类别的图像数据。这些特性都使得物体认知系统拥有了更好的多角度多场景下的物体识别能力。图 5-6 以耳机的识别为例，展示了物体认知系统的多角度物体识别能力。

本系统可以适应多种任务场景。本系统的设计宗旨是以低代价为用户提供个性化的解决方案。用户可以通过上传标注图片的方式来自定义他想使用物体认知系统解决

的问题。为了展示本章实现的物体认知系统具备的灵活性，以及呈现一些本系统的适用场景，本节设计了一系列任务，如图 5-7 所示。

图5-6 多角度下的物体识别

5.4 系统资源占用分析

本节探讨本文所提出的物体认知系统在嵌入式设备上部署的可行性。本章所展示的物体认知系统以手机应用作为移动端的软件载体，但在一些场景下，需要使用其它嵌入式设备作为移动端。例如，在一些以终端硬件为销售载体的场景中，比如智能家具、智能玩具等，都需要将数据采集，甚至数据展示的功能交由嵌入式硬件实现。本章所实现的物体认知系统也可以使用嵌入式设备作为移动端，以适应这些的应用场景。如 3.3.2 节所述，本章所使用的特征提取模型可以被部署在搭载着 Quad Core 1.2GHz Broadcom BCM2837 64bit CPU 以及 1GB RAM 的 Raspberry Pi 3 Model B 上并达到秒级的每帧图像处理速度。当所选用的嵌入式设备运算能力不足，导致响应时间过长时，可以使用第四章所提出的推理过程切分算法，用网络设备与服务器分担特征提取过程的运算，以加快响应速度。值得指出的是，第三章所提出的可增量图像识别框架与第四章所提出的多层神经网络推理过程切分算法是兼容的，可以被同一套物体认知系统所使用。第三章所提出的框架的核心是对于模型的切分，它将模型切分为固定的特征提取部分与可变的增量学习部分。而第四章着眼于将推理运算分配到各设备的方法。当这两种技术被联合使用时，模型仍会被清晰的分为固定的与可变的两部分，但是这道分界线不再会影响到推理运算在物理设备中的分配。

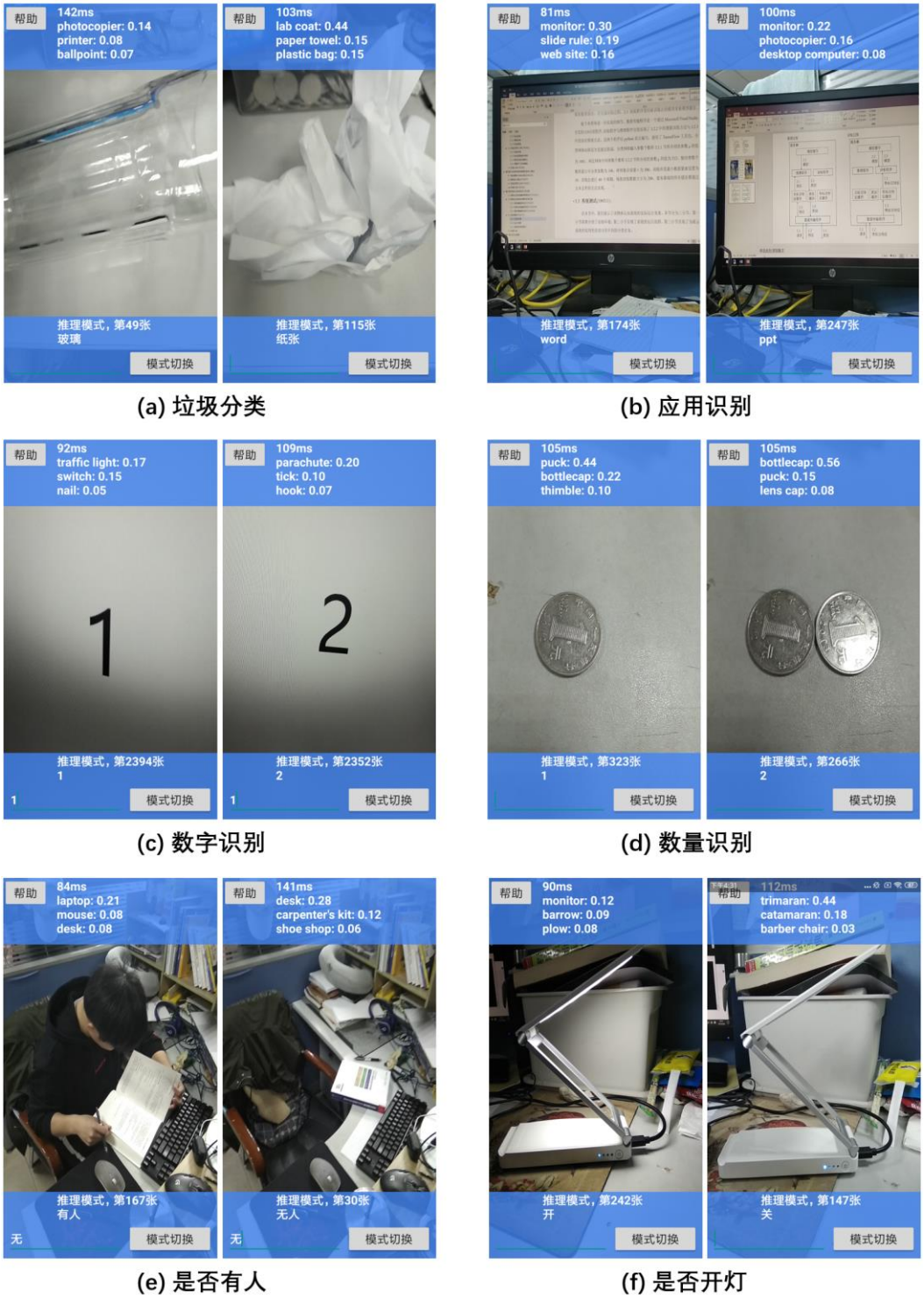


图5-7 适用场景

以嵌入式设备为移动端部署特征提取过程的问题其实等价于在嵌入式设备上执行深度学习运算推理过程的问题，而后者一直是近年来的研究热点。本章系统所依赖的 TensorFlow Lite[51]开发框架与 MobileNet 模型[24]就是科研界与产业界工作者们针

对这个问题所产生的研究成果。更多的相关工作在 2.3.3 节中也有所阐述。在此专门针对在嵌入式设备上执行深度学习运算推理过程的问题做一个回顾。首先，为了获得一个大致的印象，先来看一个案例。在 Sun 等人[50]的工作中，在 Zuluko 片上系统中部署了 SqueezeNet 模型的推理运算来对长宽均为 227 的 RGB 图像进行识别操作，平均的识别时间是 320 毫秒，而平均的内存占用是 10 MB。其中的 Zuluko 搭载了频率为 1 GHz 的 ARM v7 内核与 512 MB 的 RAM。他们使用的框架是 ARM Compute Library。接下来，从深度学习模型的资源需求，与如今各类嵌入式设备所能提供的资源入手，本节阐述了当前这一领域的研究现状。

用于图像识别的深度学习模型通常有较高的资源需求，同时也存在着压缩模型的研究。例如，AlexNet [35]的模型约有 60 M 个参数，所占用的存储空间约为 240 MB。更大的模型通常能带来更高的准确率。为了追求更高的准确率，近年来一部分研究者所提出的模型也越来越庞大。例如，VGG[58]占用的存储空间约为 527 MB。然而，也存在着压缩模型的研究，例如本文所使用的 MobileNet[24]，它的模型大小可以调节，拥有约 0.47 M 到 4.24 M 参数，其推理过程约需 14 M 到 569 M 次乘加运算。Iandola 等人[59]提出的 SqueezeNet 则声称能将模型压缩到 0.5 MB 并保持与 AlexNet 相同级别的准确率。

如今的嵌入式处理器大多使用 ARM 内核。ARM 内核共分为三类，分别是适用与微控制器的 Cortex-M 系列，适用于高性能实时系统的 Cortex-R 系列，适用于高端应用处理器的 Cortex-A 系列。Cortex-M 系列是典型的低功耗处理器，其常见用法是集成存储器、时钟与各类外设后作为微控制器使用。有很多基于 Cortex-M 系列内核的微控制器，举个例子，飞思卡尔出的 MKL25Z128 使用 Cortex-M0+内核，拥有 128 KB 的 Flash，CPU 最高频率为 48MHz。恩智浦出的 S32K 系列，使用 Cortex-M4F/M0+内核，支持最高 2 MB 的 Flash，CPU 最高频率可达 112MHz。这些微控制器被广泛用于诸如智能燃气表等低成本，低功耗的场景下。Cortex-R 系列是衍生品体量最小的 ARM 处理器，它针对高性能实时应用，例如硬盘控制器、网络设备、以及汽车应用等。就其性能来说，Cortex-R4 主频可以高达 600 MHz，Cortex-R7 的主频可超过 1 GHz。值得注意的是，虽然 Cortex-R 系列性能较高，但以它们为内核的嵌入式芯片仍不适合运行较复杂的操作系统，例如 Linux 和 Android，需要使用这些操作系统的应

用场景通常会选择 Cortex-A 处理器。Cortex-A 系列为利用复杂操作系统的设备提供了一系列解决方案，这包括了从低成本手持设备到机顶盒、平板电脑、智能手机等各类设备。本文做实验所使用的树莓派，其型号为 Raspberry Pi 3 Model B，所搭载的 BCM2837 芯片就使用了 4 颗 Cortex-A53 内核。而本章所使用的红米手机，其型号为 Redemi Note 4X，所搭载的高通骁龙 625 处理器也配备了 8 颗 Cortex-A53 内核。

回到本文所做的工作中，本文选用了基于 Cortex-A 系列处理器的嵌入式设备进行实验。其主要原因在于，尽管经过了大量简化，但一个实用的，用于计算机视觉的深度学习模型仍然较为庞大，其计算量也较大。这与 Cortex-M 系列所针对的低成本，低功耗应用场景不符合。此外，还有软件支持的问题。为简化实现过程，深度学习模型通常依赖于各类框架，然而这些框架往往运行在一个较为复杂的操作系统上。综上所述，本章进行了这样的硬件选型。然而由于 Cortex-M 系列微控制器的低成本特性，在某些场合下，仍然有在这些微控制器上部署基于图像识别的物体认知系统的需求。针对这一需求，本文的第六章设计并实现了低资源嵌入式设备上的物体认知系统。

5.5 本章小结

本物体认知系统最大的优势在于持续学习的能力以及快速的再训练。本系统的设计宗旨是以低代价为用户提供个性化的解决方案，这就要求系统随时具备快速学习用户提供的新知识的能力。本系统使用了前文提出的可增量图像识别框架，其中应用了增量学习技术使得系统能随时学习用户给出的新知识。而冻结的特征提取网络大大降低了再次训练所需的代价。本章实现了该物体认知系统，呈现出了它实际运行的效果，并以若干样例任务直观的展示了该系统能在多种场景下发挥作用。

第六章 低资源嵌入式设备上的实现

在 5.4 节中罗列了现有的用于计算机视觉的深度学习模型所需的资源，以及如今的嵌入式设备所能提供的资源。从中可以看出以 Cortex-M 系列为内核的微控制器无法支撑深度学习推理过程。然而，出于成本等方面的考虑，仍存在着在这些微控制器上部署基于图像识别的物体认知系统的需求。针对这一问题，本章设计了一种极低资源消耗的图像识别算法。本章对该算法进行了测试，并给出了它在 Cortex-M 系列嵌入式设备上的实现。

6.1 算法设计

本章提出的低资源消耗图像识别算法仍然遵照第三章的框架，算法可分为特征提取、增量训练、以及实时推理三个部分。在本章中这三个部分被大幅简化，使得所有的这三个部分可以被部署在一块资源极少的 Cortex-M 系列嵌入式设备中。

6.1.1 特征提取

如 3.2.1 节所阐述，特征提取部分的意义在于压缩数据，并使得压缩后的特征仍包含分类所需的信息。前文的实验中以预训练得到的深度学习模型作为特征提取模块。然而，如 5.4 节所示，如今用于图像识别的深度学习模型有较高的资源需求，无论是存储需求还是运算需求都超出了 Cortex-M 系列嵌入式设备所能承受的范围。为此，本节使用一种简化版的特征提取方法。该特征提取方法是将图像均分成若干块，然后求各块的灰度均值。图 6-1 展示了使用这种特征提取方法，将图像压缩为长度为 25

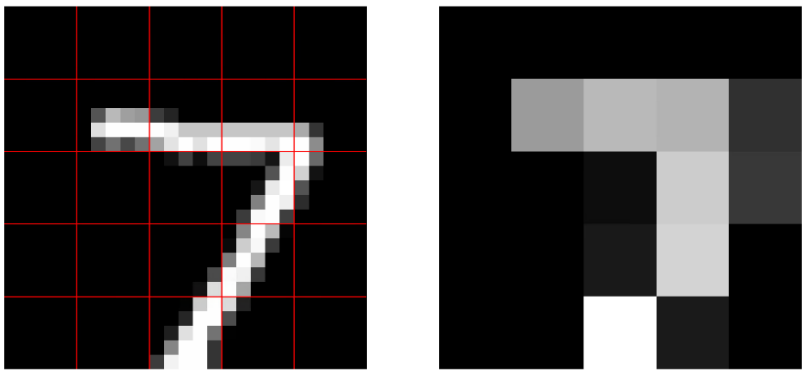


图6-1 特征提取方法

个字节的特征后的效果。该特征提取方法只需要遍历一次图像就能提取出特征，且除去存储图像与特征所需的空间外，几乎没有其余的空间消耗。

6.1.2 增量训练

物体认知系统与传统图像识别系统的一大区别就是物体认知系统存在增量训练的过程。本章的增量训练方法的接口与 3.1.3 节中的式（3.2）相同。此时的模型被定义为$\mathrm{M}=\{m^1,...,m^n\}$。其中的模版$m$有着与特征$f$相同的数据结构，而$n$代表了已经习得的类别数量。$f^y$代表被标注为$y$的特征，其中$y\in 1,...,n+1$。若$y$等于$n+1$，则表明$f^y$属于模型中没有的类别，此时在$\mathrm{M}$中新增一个类别$m^{n+1}$，并使得$m^{n+1}=f^y$。若$y$小于$n+1$，则表明$f^y$属于模型中已有的类别，此时使用

$$m^y = m^y(r-1)/r + f^y/r \tag{6.1}$$

更新模型，式中的r是人为设置的学习速率参数，r越大则模型更新越慢。该公式意味着将新获取到的带标注特征加权后汇入模型内该标注对应的类别中。

6.1.3 实时推理

实时推理过程的接口与 3.1.3 节中的式（3.3）相同，具体的实现方法为

$$y = \arg\min_{i\in 1,...,n}(\| m^i - f \|) \tag{6.2}$$

即y为M中与待分类特征f距离最近的模版所属的类别。式中的$\| m^i - f \|$代表待分类特征与M中各模板特征之间的距离。式中的 argmin 代表取最小值对应的类别i。

6.2 算法测试

上一节阐述了算法的实现细节，可以看出其资源占用极少，对资源占用的定量分析将在 6.4 节详细阐述，这一节针对算法的分类准确率进行实验。考虑到本章使用了一个非常基础的图像识别算法，本节使用 MNIST [60]手写数字数据集进行实验。MNIST 是一个图像识别入门级数据集，包含了具有 60,000 个示例的训练集和具有 10,000 个示例的测试集。与 3.3.4 节类似，本实验仍然采用增量训练的方法，实验中特征的大小为 169 像素，即 13 乘 13，每个像素用一个浮点数表示。学习速率r被设

置为 1000。实验结果如图 6-2 所示。横坐标为已习得的类别数量，纵坐标为分类准确度。当学习完所有的 10 个类别后，分类准确率为 0.814。同时，图 6-3 展示了学习完所有 10 个类别后的模型 M 中的模版。分类即从这些模版中选取与从待分类图像中提取出的特征距离最近的模版的过程。

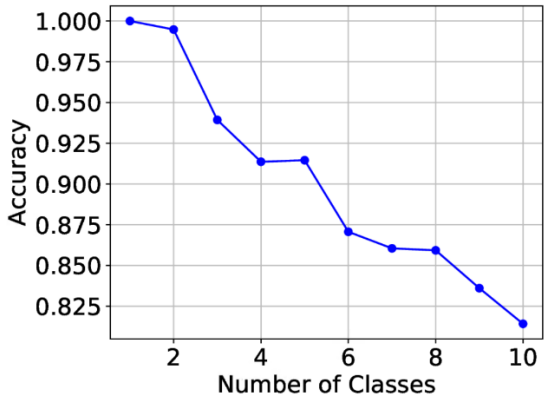


图6-2 实验准确率

6.3 系统设计

本章在 Cortex-M 系列嵌入式设备上部署了基于图像识别的物体认知系统。特征提取、增量训练、实时推理这三个模块都在同一块芯片上实现。本节将从硬件设计与软件设计两部分入手，详细阐述这套系统。

6.3.1 硬件设计

图 6-4 展示了本章实现的物体认识系统使用的硬件设备。硬件设备由主控板，摄像头，按钮，液晶屏四个部分组成。系统以两个按钮以及摄像头作为输入设备，以主控板板载三色灯以及液晶屏作为输出设备。主控板选用了苏州大学嵌入式与物联网实验室出品的 S32K 核心板以及 AHL-IoT-GEC 扩展板，主控板上搭载了 S32K144 微控制器，是整套系统的控制核心。摄像头选用了上海骑远飞电子科技出品的 QYF-BWSC03 串口摄像头模块，与主控板之间通过 UART 通信。液晶屏与主控板之间主要通过 SPI 进行通信，同时主控板还可以通过 GPIO 引脚对液晶屏进行复位等状态设

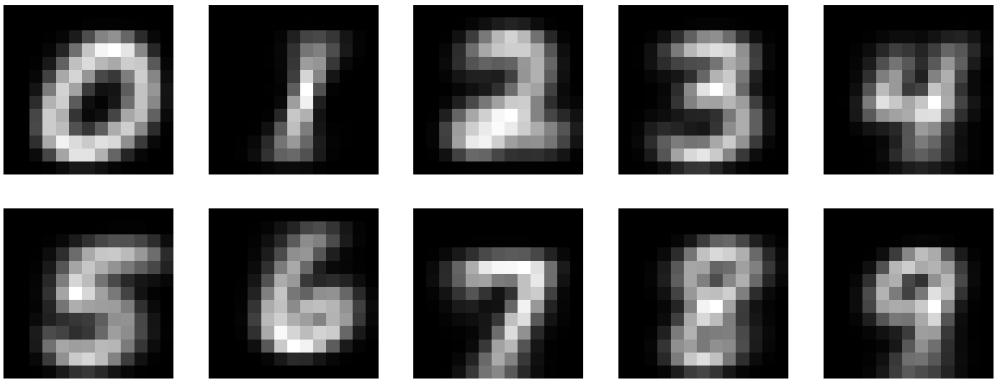


图6-3 训练得到的模型

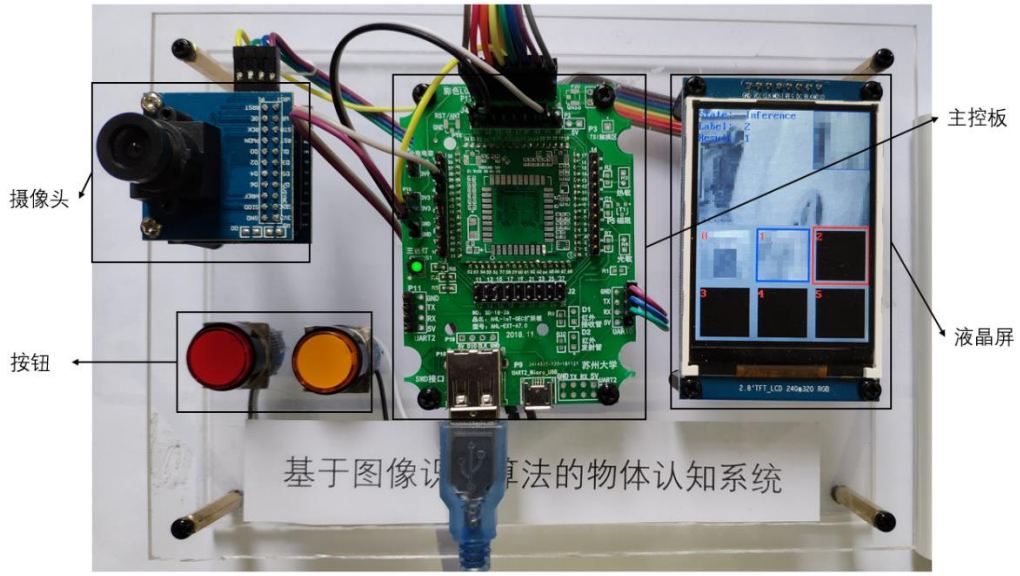


图6-4 系统硬件设备

置。按钮与主控板之间通过 GPIO 通信。

6.3.2 软件设计

系统实现了 6.1 节所阐述的算法，在此基础上，为了实现与用户的交互，设计并实现了一套运行流程。图 6-5 展示了系统的运行流程。系统共存在三个运行状态，分别是，设定标注状态、推理状态与训练状态。通过按钮可以进行状态切换以及设定标注，共有两个按钮，分别是标注按钮与训练按钮。

设定标注状态是为了让用户设定标注，当系统运行在设定标注状态下时，实验现象为三色灯常亮，显示标注所对应的颜色。除去黑白外，三色灯共能显示六种颜色，因此本系统设计为最多容纳六个类别，分别对应的颜色为 0 红，1 绿，2 蓝，3 青，4 品红，5 黄。在设定标注状态下按下标注按钮，将循环更改标注的类别为下一个类别。在设定标注状态下按下训练按钮，将切换到推理状态。

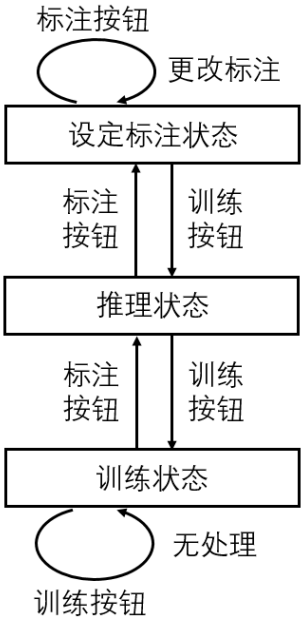


图6-5 系统状态转换图

当系统运行在推理状态下时，系统将使用模型对摄像头拍摄到的图像进行分类。此时的实验现象为三色灯色黑闪烁，闪烁的颜色为分类结果所对应

的颜色。在推理状态下按下标注按钮，将切换到标注状态。在推理状态下按下训练按钮，将切换到训练状态。

当系统运行在训练状态下时，系统将用摄像头拍摄到的图像以及用户给出的标注训练模型。此时的实验现象为三色灯色白闪烁，颜色为标注的颜色。在训练状态下按下标注按钮，将切换到推理状态。在训练状态下按下训练按钮，系统不作处理。

为了让用户能更直观的了解系统内部的情况，除了三色灯以外，本系统还使用液晶屏进行结果输出，如图 6-6 所示。屏幕的上半部分显示摄像头拍摄到的画面，其中左上角会以文本的形式显示运行状态、标注以及分类结果，右上角显示从图片中提取出的特征。屏幕的下半部分则显示模型的状态。

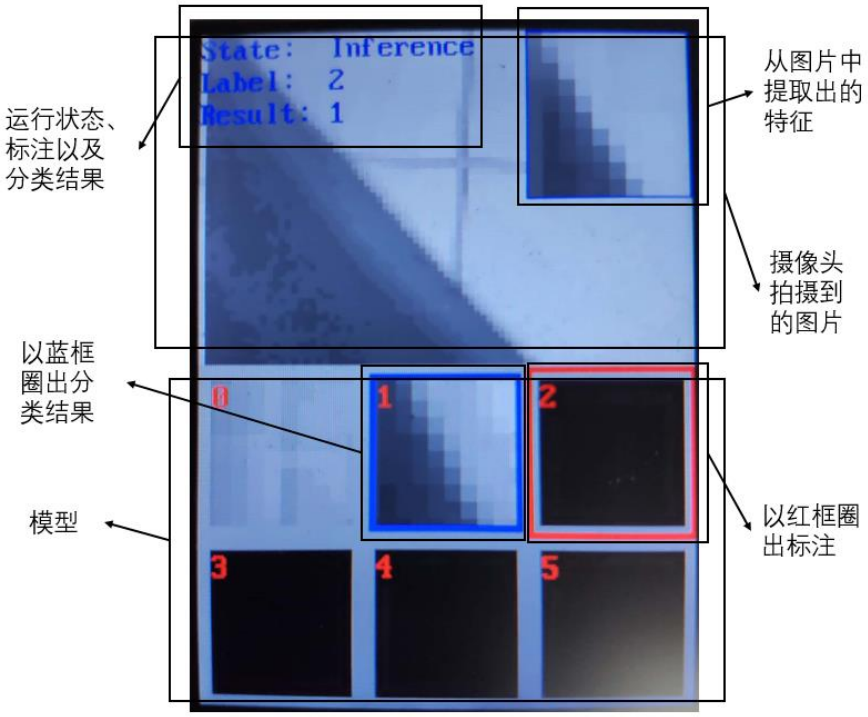


图6-6 系统硬件设备

6.4 系统测试

与第五章类似，本系统遵循着上电，标注，训练，推理的典型运行流程。本节主要针对该系统的存储资源占用，运行速度等方面对该系统进行分析与测试。

本系统对于存储资源占用有着苛刻的要求。本系统使用恩智浦出品的 S32K144 微控制器。这款微控制器搭载着 Cortex-M4 内核，拥有 512KB 的 Flash 以及 60KB 的

RAM。就 Falsh 占用而言，编译完成的程序大小为 13.0KB，在运行过程中不会产生其他 Flash 占用。就 RAM 占用而言，系统运行中主要需要申请三个较大的静态变量，它们分别用于存放从摄像头读取到的图像，从图像中提取出的特征，以及模型。图像使用单字节整形存储一个灰度像素，图像高度为 60 像素，图像宽度为 80 像素，存放一幅图像共需要 4.8KB 的 RAM。特征使用四字节浮点型存放每个像素，特征的高度与宽度都为 10 像素，存放一幅特征需要 0.4KB 的 RAM。模型要为每个类型保留一幅特征，6 个类别，每个类别 0.4KB，共需 2.4KB。在系统运行的过程中，内存中需要驻留一幅图像，一幅特征，一份模型，共占用 7.6KB 的 RAM。

就实时性而言，该系统可以达到秒级的处理速度。经测定，系统在推理状态下运行时的帧率为 0.61 帧每秒，在训练状态下运行时的帧率为 0.59 帧每秒。

6.5 本章小结

针对资源严格受限的嵌入式设备，例如以 Cortex-M 系列为内核的微控制器，本章以第三章所提出的框架为基础，大幅削减了模型的复杂度，设计并实现了更低资源占用的物体认知系统。本章在 MNIST 数据集上验证了该算法，得到了 0.814 的分类准确率。本章还给出了该系统在 S32K144 上的实现，经过分析与测定，该系统仅占用了 13.0KB 的 Falsh 与 7.6KB 的 RAM，且能达到秒级的处理速度。本章所做的工作扩充了本文提出的框架的适用范围，同时也很好的说明该框架是可扩展的，是独立于特定算法与任务的。

第七章 总结与展望

7.1 总结

与传统的基于射频识别等传感技术的物体认知系统不同，本文提出的物体认知系统是一种部署在物联网环境中的，具备持续学习能力的图像识别系统，是人工智能与物联网的结合。人工智能技术与物联网的结合一直是国内外研究的热点。针对目前智能物联网中存在的如持续学习能力缺失，运算分配不合理等一系列问题。本文引入了增量学习、迁移学习、边缘计算等一系列研究成果，对物体认知系统的整体框架设计、运算分配方案、持续学习方法、部署及运行流程、用户交互方法等关键技术进行了研究。总结如下：

（1）针对物体认知系统的图像处理能力，提出了适用与物联网环境的可增量图像识别算法框架。该框架将增量学习方法引入到智能物联网应用中，并针对物联网环境对图像识别模型进行了分割。实验证明该框架能有效降低数据传输量，快速的完成训练与推理过程，并且具有良好的并发处理能力以及持续学习能力。

（2）针对物体认知系统的运算分配问题，提出了多层神经网络推理过程在多层物联网架构下的切分算法。本文建立了深度学习推理过程在多层物联网架构下的运算分配模型，并在该模型下以最小化响应时间为目标提出了一种多层神经网络推理过程切分算法。该算法能在线性时间复杂度内计算得到最优分配策略。本文通过在现实设备中进行的实验验证了该方案的可行性与有效性。

（3）基于上述技术，设计并实现了一款基于图像识别技术的物体认知系统。本系统是上述可增量图像识别算法框架在实际中的具体应用，能在部署后持续的学习用户所教授的知识。这是一套高自由度，广适应性的系统，能以低代价为用户提供个性化的解决方案。本文实现了这套物体认知系统，通过若干实验直观的展示了系统运行效果，并罗列了若干可以应用该系统的场景。

（4）针对将物体认知系统部署在低资源嵌入式设备上的问题，本文在上述可增量图像识别算法框架的基础上进行了大量简化，提出了一种极低资源消耗的基于图像的物体认知算法。本文对该算法进行了测试，并给出了它在 S32K144 上的实现。实

验表明，即使在这类资源极其受限的嵌入式设备上，也能部署基于图像识别的物体认知系统。

7.2 展望

本文对可增量图像分类框架，推理过程分配策略等物体认知系统关键技术做了研究，并实现了物体认知系统。但是仍然存在一些问题，有待今后进一步研究与实现：

（1）可增量图像识别算法框架方面，需要研究该框架的扩展能力。当前该框架适用于普通的图像分类应用，然而在一些情况下，物联网图像处理应用需要具备针对某一领域的高准确率，或者需要具备物体定位等其他能力。将本框架延伸到这些领域仍是值得探索的方向。

（2）多层神经网络推理过程切分算法方面，需要考虑功耗问题以及研究算法输入数据测量方法。很多的物联网设备，尤其是电池供电的终端设备都有低功耗的要求。对于功耗指标的考虑能够扩展该模型，使它更适用于这些要考虑低功耗的场景。此外，算法的一系列输入数据是测量得到的，对于多任务并发环境、动态的物联网运行环境来说，如何准确而低代价的测得这些输入数据仍然是值得探索的方向。

（3）物体认知系统方面，需要进行多用户处理，优化性能等操作。本文实现的物体认知系统是单用户的，没有对多用户的情况进行处理。增加这些内容可以增强它的实用性。

参考文献

[1] Al-Fuqaha A, Guizani M, Mohammadi M, et al. Internet of Things: A Survey on Enabling Technologies, Protocols and Applications [J]. IEEE Communications Surveys & Tutorials, 2015, 17(4): 2347-2376.

[2] 杨震, 杨宁, 徐敏捷. 面向物联网应用的人工智能相关技术研究 [J]. 电信技术, 2016, 1(5): 16-19.

[3] Wang Y P E, Lin X, Adhikary A, et al. A Primer on 3GPP Narrowband Internet of Things [J]. IEEE Communications Magazine, 2017, 55(3): 117-123.

[4] Cherukutota N, Jadhav S. Architectural framework of smart water meter reading system in IoT environment [A]. 2016 International Conference on Communication and Signal Processing (ICCSP) [C]. Piscataway, NJ: IEEE, 2016: 0791-0794.

[5] 伦一. 自动驾驶产业发展现状及趋势 [J]. 电信网技术, 2017, (6): 40-43.

[6] Lecun Y, Bengio Y, Hinton G. Deep learning [J]. Nature, 2015, 521(7553): 436.

[7] Hayes B. Cloud computing [J]. Communications of the Acm, 2008, 51(7): 9-11.

[8] Walker S J. Big Data: A Revolution That Will Transform How We Live, Work, and Think [J]. Mathematics & Computer Education, 2014, 47(17): 181-183.

[9] Maulik U, Lawrence B, Diane J, et al. Advanced methods for knowledge discovery from complex data [M]. Berlin: Springer, 2006.

[10] Bonomi F, Milito R, Zhu J, et al. Fog Computing and Its Role in the Internet of Things [A]. Proceedings of the first edition of the MCC workshop on Mobile cloud computing [C]. New York: ACM, 2012: 13-16.

[11] Shi W, Cao J, Zhang Q, et al. Edge Computing: Vision and Challenges [J]. IEEE Internet of Things Journal, 2016, 3(5): 637-646.

[12] Lane N D, Bhattacharya S, Georgiev P, et al. An Early Resource Characterization of Deep Learning on Wearables, Smartphones and Internet-of-Things Devices [A].

Proceedings of the 2015 International Workshop on Internet of Things Towards Applications [C]. New York: ACM, 2015: 7-12.

[13] Bhattacharya S, Lane N D. Sparsification and Separation of Deep Learning Layers for Constrained Resource Inference on Wearables [A]. Proceedings of the 14th ACM Conference on Embedded Network Sensor Systems CD-ROM [C]. New York: ACM, 2016: 176-189.

[14] He L, Ota K, Dong M. Learning IoT in Edge: Deep Learning for the Internet of Things with Edge Computing [J]. IEEE Network, 2018, 32(1): 96-101.

[15] Li E, Zhou Z, Chen X. Edge Intelligence: On-Demand Deep Learning Model Co-Inference with Device-Edge Synergy [A]. Proceedings of the 2018 Workshop on Mobile Edge Communications [C]. New York: ACM, 2018: 31-36.

[16] Teerapittayanon S, Mcdanel B, Kung H T. Distributed Deep Neural Networks over the Cloud, the Edge and End Devices [A]. 2017 IEEE 37th International Conference on Distributed Computing Systems (ICDCS) [C]. Piscataway, NJ: IEEE, 2017: 328-339.

[17] 顾会光, 王宜怀, 史新峰. 数据无损的远程代码更新的设计与研究 [J]. 计算机工程与设计, 2015, (10): 2633-2639.

[18] Song M, Zhong K, Zhang J, et al. In-Situ AI: Towards Autonomous and Incremental Deep Learning for IoT Systems [A]. 2018 IEEE International Symposium on High Performance Computer Architecture (HPCA) [C]. Piscataway, NJ: IEEE, 2018: 92-103.

[19] Shokri R, Shmatikov V. Privacy-Preserving Deep Learning [A]. Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security [C]. New York: ACM, 2015: 1310-1321.

[20] 熊伟. 面向移动设备的深度学习部署运算优化技术 [J]. 电子制作, 2017, (12): 92-94.

[21] Han S, Pool J, Tran J, et al. Learning both Weights and Connections for Efficient Neural Networks [A]. Advances in Neural Information Processing Systems [C]. New York: ACM, 2015: 1135-1143.

[22] Gupta S, Agrawal A, Gopalakrishnan K, et al. Deep Learning with Limited Numerical Precision [A]. International Conference on Machine Learning [C]. New York: ACM, 2015: 1737-1746.

[23] Lavin A, Gray S. Fast algorithms for convolutional neural networks [A]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition [C]. Piscataway, NJ: IEEE, 2016: 4013-4021.

[24] Howard A G, Zhu M, Chen B, et al. MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications [EB/OL]. https://arxiv.org/abs/1704.04861, 17 Apr 2017.

[25] French R M. Catastrophic forgetting in connectionist networks [J]. Trends in Cognitive Sciences, 1999, 3(4): 128-135.

[26] Kirkpatrick J, Pascanu R, Rabinowitz N, et al. Overcoming catastrophic forgetting in neural networks [J]. Proceedings of the National Academy of Sciences, 2017, 114(13): 3521-3526.

[27] Li Z, Hoiem D. Learning Without Forgetting [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018, 40(12): 2935-2947.

[28] Rebuffi S A, Kolesnikov A, Sperl G, et al. iCaRL: Incremental Classifier and Representation Learning [A]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition [C]. Piscataway, NJ: IEEE, 2017: 2001-2010.

[29] 孟小峰, 慈祥. 大数据管理:概念、技术与挑战 [J]. 计算机研究与发展, 2013, 50(1): 146-169.

[30] Doersch C, Gupta A, Efros A A. Unsupervised Visual Representation Learning by Context Prediction [A]. Proceedings of the IEEE International Conference on Computer Vision [C]. Piscataway, NJ: IEEE, 2015: 1422-1430.

[31] Jenni S, Favaro P. Self-supervised feature learning by learning to spot artifacts[A]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition [C]. Piscataway, NJ: IEEE, 2018: 2733-2742.

[32] Shao L, Zhu F, Li X. Transfer Learning for Visual Categorization: A Survey [J]. IEEE Transactions on Neural Networks and Learning Systems, 2015, 26(5): 1019-1034.

[33] Zamir A, Sax A, Shen W, et al. Taskonomy: Disentangling Task Transfer Learning [A]. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition [C]. Piscataway, NJ: IEEE, 2018: 3712-3722.

[34] Bengio Y, Courville A, Vincent P. Representation Learning: A Review and New Perspectives [J]. IEEE Transactions on Pattern Analysis & Machine Intelligence, 2012, 35(8): 1798-1828.

[35] Krizhevsky A, Sutskever I, Hinton G. ImageNet Classification with Deep Convolutional Neural Networks [J]. Communications of the ACM, 2017, 60(6): 84-90.

[36] Yosinski J, Clune J, Bengio Y, et al. How transferable are features in deep neural networks? [A]. Advances in neural information processing systems [C]. Cambridge, MA: MIT Press, 2014: 3320-3328.

[37] 王保云. 物联网技术研究综述 [J]. 电子测量与仪器学报, 2009, 23(12): 1-7.

[38] Want R. An introduction to RFID technology [J]. IEEE Pervasive Computing, 2006, 5(1): 25-33.

[39] Yick J, Mukherjee B, Ghosal D. Wireless sensor network survey [J]. Computer Networks, 2008, 52(12): 2292-2330.

[40] 魏立明, 吕雪莹. 物联网技术研究综述 [J]. 数码世界, 2016, (8): 36-37.

[41] Biswas A R, Giaffreda R. IoT and cloud convergence: Opportunities and challenges [A]. 2014 IEEE World Forum on Internet of Things (WF-IoT) [C]. Piscataway, NJ: IEEE, 2014: 375-376.

[42] 陈全, 邓倩妮. 云计算及其关键技术 [J]. 计算机应用, 2009, 29(9): 2562-2567.

[43] 程学旗, 靳小龙, 王元卓, et al. 大数据系统和分析技术综述 [J]. 软件学报, 2014, 25(9): 1889-1908.

[44] 童晓渝, 房秉毅, 张云勇. 物联网智能家居发展分析 [J]. 移动通信, 2010, 34(9): 16-20.

[45] Tang J, Sun D, Liu S, et al. Enabling Deep Learning on IoT Devices [J]. Computer, 2017, 50(10): 92-96.

[46] De V, Borkar S. Technology and design challenges for low power and high performance [microprocessors] [A]. Proceedings. 1999 International Symposium on Low Power Electronics and Design (Cat. No. 99TH8477) [C]. Piscataway, NJ: IEEE, 1999: 163-168.

[47] Hahm O, Baccelli E, Petersen H, et al. Operating Systems for Low-End Devices in the Internet of Things: a Survey[J]. IEEE Internet of Things Journal, 2016, 3(5): 720-734.

[48] Cheng K T, Wang Y C. Using mobile GPU for general-purpose computing – a case study of face recognition on smartphones [A]. Proceedings of 2011 International Symposium on VLSI Design, Automation and Test [C]. Piscataway, NJ: IEEE, 2011: 1-4.

[49] Soyata T, Muraleedharan R, Funai C, et al. Cloud-Vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture [A]. 2012 IEEE symposium on computers and communications (ISCC) [C]. Piscataway, NJ: IEEE, 2012: 000059-000066.

[50] Sun D, Liu S, Gaudiot J L. Enabling Embedded Inference Engine with ARM Compute Library: A Case Study [EB/OL]. https://arxiv.org/abs/1704.03751, 14 Apr 2017.

[51] Abadi M, Barham P, Chen J, et al. Tensorflow: A system for large-scale machine learning [A]. OSDI'16:Proceedings of the 12th USENIX conference on Operating Systems Design and Implementation [C]. Berkeley, CA: USENIX Association, 2016: 265-283.

[52] Judd P, Albericio J, Hetherington T, et al. Reduced-Precision Strategies for Bounded Memory in Deep Neural Nets [EB/OL]. https://arxiv.org/abs/1511.05236, 8 Jan 2016.

[53] Russakovsky O, Deng J, Su H, et al. ImageNet Large Scale Visual Recognition Challenge [J]. International Journal of Computer Vision, 2015, 115(3): 211-252.

[54] Lomonaco V, Maltoni D. CORe50: a New Dataset and Benchmark for Continuous Object Recognition [EB/OL]. https://arxiv.org/abs/1705.03550, 9 May 2017.

[55] Kang Y, Hauswald J, Gao C, et al. Neurosurgeon: Collaborative Intelligence Between the Cloud and Mobile Edge [J]. Acm Sigplan Notices, 2017, 52(4): 615-629.

[56] 范国亮. 窄带物联网技术及其市场趋势 [J]. 中国新通信, 2017, 19(19): 117.

[57] 李敏. 是什么让 5G 的峰值速度高达 20Gb/s [J]. 计算机与网络, 2017, 43(10): 53.

[58] Simonyan K, Zisserman A. Very deep convolutional networks for large-scale image recognition [EB/OL]. https://arxiv.org/abs/1409.1556, 10 Apr 2015.

[59] Iandola F N, Han S, Moskewicz M W, et al. SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size [EB/OL]. https://arxiv.org/abs/1602.07360, 4 Nov 2016.

[60] Lecun Y L, Bottou L, Bengio Y, et al. Gradient-Based Learning Applied to Document Recognition [J]. Proceedings of the IEEE, 1998, 86(11): 2278-2324.

攻读硕士学位期间主要的研究成果

[1] Zhou J, Wang Y, Ota K, et al. AAIoT: Accelerating Artificial Intelligence in IoT Systems [J]. IEEE Wireless Communications Letters, 2019, 8(3): 825-828. (SCI 二区)

[2] 参加横向科研项目 P11801617《燃气表软硬件设计》.

致 谢

时光如白驹过隙，转眼间就到了研究生生涯的尾声。这也是我在苏州大学学习的第七个年头，心中无限感慨，既有对我深爱的母校难以割舍的情怀，又有对未来那美好的憧憬。此刻，我真的很希望时间能够慢点走，让我再次漫步在林荫小道，让我再次聆听朗朗的读书声，让我再次闻闻那沁人心脾的桂花香。

我要感谢我的父母们。我要感谢他们含辛茹苦将我养大成人，感谢他们为我创造优越的生活条件，感谢他们以身作则教会我为人处事的道理。他们在我的背后默默地支持我，在我有困难的时候陪伴在我的身边。我会用我的双手和汗水来回报他们。

我要感谢我的恩师姚望舒教授、王宜怀教授、以及董冕雄教授。在我三年的研究生学习中给予了我很多帮助，是你们教我方法论并以自己严谨的科研态度和强烈的社会责任感时刻的感染着我，使我坚定自己一定要做个对社会有贡献的人。我真的很幸运遇到了这样好的老师们，不仅在学习上提供指导，而且在为人处事上也教会了我很多道理。

我要感谢我的同学们。感谢他们在生活和学习上对我的照顾和帮助。我为自己是苏州大学嵌入式实验室这个大家庭的一员感到骄傲与自豪，在这个集体内我感觉到很快乐、很开心。在此，感谢程宏玉、张晁逢、李良知、陈成、胡夏禹、钱涵佳、汪博等对本论文提出了宝贵的修改意见。怀念大家一起在实验室探索学习的感觉。

最后，感谢各位评审专家对我论文的审阅，衷心感谢各位评委、老师的批评指正。

周靖越

二〇一九年三月十七日